tanulmányok

87/1978

MTA Számitástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

OPERÁCIÓS RENDSZEREK ELMÉLETE
IV. VISEGRÁDI TÉLI ISKOLA

ISBN 963 311 074 2

Tanulmányok 87/1978.

Felelős kiadó:

Dr VÁMOS TIBOR

MTA Számítástechnikai és Automatizálási Kutató Intézet
MTA Számítástudományi Bizottsága

A konferenciát a "Számítástechnika tudományos kérdései" c.
többoldalú akadémia együttmüködés keretében rendeztük.

Конференция была проведена в рамках
многостороннего сотрудничества академий
социалистических стран по проблеме
"Научные вопросы вычислительной техники"

Conference was held in the frame of the multilateral cooperation
of the academies of sciences of the socialist countries on Computer Sciences.

TARTALOMJEGYZÉK

SCHEDULING TO MEET DEADLINES WITH RESOURCE CONSTRAINTS - COMPLEXITY RESULTS

Jacek Błazewicz, Jan Weglarz
Instytut Automatyki, Politechnika Poznanska

ABSTRACT:

The problem to be considered is one of the nonpreemptive scheduling of tasks on processors, when every task requires for its execution additional resource and has associated with it an execution time, ready time and a deadline before which must be completed. For the simplest case of scheduling unit length, independent tasks on two processors with one unit of one resource type available in the system, an efficient algorithm for finding a schedule with no task late, whenever such a schedule exists, is presented. If we allow, in the last problem, for an arbitrary number of resource types available in the system, it is unknown whether or not there exists an efficient algorithm for solving this new problem. Other problems are proved to be NP-complete, hence, computationally intractable. Moreover, a method of solving the the general scheduling problem with unit length, independent tasks, by the reduction to the problem of network flow with multipliers, is developed.

1. INTRODUCTION

The problem to be considered is one of scheduling tasks on processors to meet deadlines. Such a case often arises in process control and monitoring systems in which the computer must guarantee that every task will be completed before some fixed time has elapsed, after which the obtained results will be useless. Such a problem has received much attention in recent years [8,13,14,16] (see [3] as a survey). However the use of additional resources was not considered in these papers. A more general and more realistic model of computing systems, when some of the tasks may require the use of various limited resources during their execution, was first examined in [9,10,12].

In [10], scheduling nonpreemptable tasks to minimize schedule length was considered. It was shown that almost all problems are NP-complete and hence they are as computationally intractable as the travelling salesman problem. In [9], bounds on scheduling with limited resources were obtained. In [12] the special case was studied in which there is only one type of additional resource which can be shared by a limited number of tasks. Lastly, in [4], the problem of scheduling to minimize mean flow time was considered. It was proved that the same problems as in the case of minimizing schedule length are Np-complete. A polynomial in time algorithm was also given for the simplest case.

In this paper, we survey results obtained in the field of scheduling tasks to meet deadlines under resource contraints [5].

We consider the following model of a computing system. There are n tasks $T_1, T_2, \ldots, T_n$, which are to be processed on m identical processors $P_1, P_2, \ldots, P_m$. There is also given a set of additional resources $\mathcal{R}=[R_1, R_2, \ldots, R_s]$. For each resource R_1 there exists a bound m_1 (integer) which gives the total amount of that resource available at any given time. For every task T_i, $i=1, 2, \ldots, n$, there are given: <u>execution time</u> τ_i, <u>ready time</u> r_i, <u>deadline</u> d_i and vector $R(T_i)$ (integer) containing resource requirements (the 1-th component of this vector $R_1(T_i) \leq m_1$, denotes the number of units of R_1 required by task T_i). A partial order $<$, specifying precedence constraints is defined on the task set. $T_i < T_j$ means that T_j cannot begib until T_i is finished.

Let us recall some definitions which will be useful in the following. For every task T_i in the schedule, we denote by C_i its <u>completion time</u>. The schedule will be called <u>optimal</u> if all task deadlines are respected in it, that is for every T_i the condition $C_i \leq d_i$ is fulfilled. A scheduling algorithm is a procedure that produces a schedule for every given set of tasks. By a <u>preemptive</u> scheduling algorithm, we mean to allow the interruption of the execution of tasks in a schedule. When interruptions are not allowed, we call an algorithm <u>nonpreemp-</u>

tive. In this paper we will be concerned with nonpreemptive scheduling algorithms only. By optimal scheduling algorithm we mean here the algorithm which finds an optimal schedule, whenever one exists.

A distinction will be made between some scheduling sub-problems which differ from each other by the values of problem parameters /that is by problem inputs/. This is due to the fact that some algorithms apply only to certain restricted classes of these inputs. The complexity of a scheduling subproblem also depends strongly on its input. For convenience, to denote such a subproblem, we adopt, with slight modification, the notation of [15] : $n|m|\lambda|\kappa$, where:

n- number of tasks;

m- number of processors;

- problem parameters such as: s=k k resource types . m_1 = r (r units of resource R_1) , < (arbitrary precedence constraints between the tasks) , forest (precedence constraints between the tasks such that the associated precedence graph forms a forest), $r_i \neq 0$ (possibly non-equal ready times for tasks) , $\tau_i = 1$ (unit processing times) , pr (preemptive algorithm), nonpr (nonpreemptive algorithm) ;
- optimality criterion, for example schedule length $C_{max} = \max_i \{C_i\}$, maximum lateness $L_{max} = \max_i \{C_i - d_i\}$.

We will be concerned with problems which are formulated in such a way an answer to them may only be "yes" or "no". As

is known [1,7,11] this is not restrictive, since every scheduling problem can be formulated in such a manner. Following this, we may denote problem of scheduling to meet deadlines by $\mathcal{K}=C_i \leqslant d_i$.

The main subject of our interest will be the computational complexity of the scheduling problems. For some problems it will be possible to give polynomial in time algorithms which can be hence used directly in operating systems. However, most of the considered problems can be proved to be NP-complete, thus for their optimal solution we can use enumerative methods such as branch and bound or dynamic programming, since no substantially better method is likely to exist. Thus, when the NP-completeness of the problem is proved, we can use approximate approaches in scheduling procedures of operating systems.

The paper is constructed in the following way. In Section 2 an optimal and polynomial in time algorithm is given for the problem $n|2|\ s=1,\ m_1=1,\ \tau_i=1,\ \text{nonpr}|\ C_i \leqslant d_i$. Section 3 deals with NP-complete problems. It is shown that problems $n|2|\ s=1,\ \text{forest},\ \tau_i=1,\ \text{nonpr}|C_i \leqslant d_i$, $n|2|\ s=1,\ m_1=1,\ <,\ \tau_i=1,\ \text{nonpr}|\ C_i \leqslant d_i$ and $n|3|s=1, \tau_i=1,\ \text{nonpr}|C_i \leqslant d_i$ are NP-complete. The computational complexity of the remaining problem $n|2|\ s=k,\ r_i \neq 0,\ \tau_i=1,\ \text{nonpr}|C_i \leqslant d_i$ is still open. In Section 4 a method for solving the general problem of scheduling unit length, independent tasks is developed. This is done by reduction to the problem of network flows with

multipliers. Finally, the problem of minimizing maximum lateness is briefly examined.

2. ALGORITHM FOR SCHEDULING TASKS ON TWO PROCESSORS

In this Section we present a simple, polynomial in time, algorithm for scheduling unit length, independent tasks on two processors when every task may require not more than a unit of one type of additional resource. That is, we consider the problem $n|2|s=1, m_1=1, \tau_i=1$ nonpr$|C_i \leq d_i$. An algorithm for scheduling tasks in this case may be described as follows.

Algorithm 1.

1° Form the list of the task in nondecreasing order of d_i. Renumber tasks according to this order. Set $t:=0$.

2° Assign task T_1 to a processor at moment t and remove it from the list.

3° Assign task T_1 to the second processor in the following manner:

a/ If $R_1(T_1)=1$, choose the first task on the list for which $R_1(T_1)=0$. If there is no such task leave the second processor idle.

b/ If $R_1(T_1)=0$, choose the first task on the list with resource requirement $R_1(T_1)=1$, if either $l=2$, or for every k, $1<k<l$, the following condition being fulfilled:

$$t+1+\lceil (k-1)/2 \rceil \leq d_k \quad [1] \qquad (1)$$

If the above condition is not fulfilled for some k, choose task T_2.

[1] $\lceil x \rceil$ denotes the smallest integer not less than x.

4° Remove the assigned tasks from the list and set t:=t+1. If there are any tasks on the list, renumber them and go to step 2°.

Let us note that in the above algorithm formula (1) is used to check whether or not it will be possible to schedule tasks optimally if task T_1 with $R_1(T_1)=1$ is assigned.
We will new prove the optimality of Algorithm 1.

Theorem 1

Algorithm 1 is optimal for the problem $n \mid 2 \mid s=1, m_1=1, \tau_i=1, \text{nonpr} \mid C_i \leq d_i$.

Proof

Let us assume that there exists an optimal schedule π and let schedule π' obtained after using Algorithm 1 differ from it. We will prove that π' is also optimal.

First, let us assume that in $\tilde{\pi}$ a different assignment is made than that following Step 3 of the algorithm. The following cases may occur.

i. Schedule $\tilde{\pi}$ is not consistent with Step 3a of Algorithm 1. Such a case may only occur when T_1 (with $R_1(T_1)=1$) is processed alone despite the fact that T_i's with $R_1(T_i)=0$

are available. It is obvious that by assigning one of these tasks to the second processor we can improve the schedule. Thus, π' is also optimal.

ii. Schedule $\tilde{\pi}$ is not consistent with Step 3b of Algorithm 1. Thus, it is the case that in $\tilde{\pi}$ at time t there are processed: T_1, for which $R_1(T_1)=0$ (cf. Steps 2 and 3b of Algorithm 1), and T_2 for which $R_1(T_2)=0$, despite the fact that condition (1) is fulfilled. Let us note that we may, without delaying any task, process T_1 in the position formerly assigned to T_2, where T_1 is the first task on the list with $R_1(T_1)=1$. This follows from the fact that there is no such T_k, $1<k<l$, for which $R_1(T_k)=1$, and hence we may begin processing T_2 at moment $t+1$, moving the beginning of T_4 to moment $t+2$, etc. Finally, in the position formerly assigned to T_1 we process T_{l-1} (or T_{l-2} if $l=1$). All of these changes do not cause the schedule to lose its optimality, because for every rask T_k, $1<k<l$, condition (1) is fulfilled.

We consider now other cases in which $\tilde{\pi}$ and π' may be different. Notice that these cases are characterized by the fact that in the optimal schedule π, T_1 is processed first and T_k after it, despite the fact that $d_k<d_1$. The following situations may thus occur.

iii. $R_1(T_1)=R_1(T_k)$. It is obvious that in this case we can always exchange tasks T_1 and T_k and such a change will maintain the on-time status of both tasks.

iv. $R_1(T_1)=1$. $R_1(T_k)=0$. If task T_i is processed in parallel with T_1 in $\tilde{\pi}$, then if:

- $d_i<d_k$ and condition (1) is fulfilled, then the obtained schedule $\tilde{\pi}$ is consistent with Algorithm 1.

/Let us note that if condition (1) was not fulfilled then schedule π could not be optimal/;

- $d_i \geq d_k$, then according to iii., replacing T_i and T_k we do not cause the schedule to lose its optimality and tasks T_1 and T_k will be processed in parallel. If T_1 is processed alone in π, then T_k may be processed in parallel on the second processor

v. $R_1(T_1)=0$, $R_1(T_k)=1$. If task T_i is processed in parallel with T_1 in π, then if

- $d_i \geq d_k$, it is possible to change the order of processing such that T_k and the task which was processed in parallel with it in π, are processed first and in their previous position tasks T_1 and T_i, and the new schedule is also optimal;
- $d_i < d_k$ and $R_1(T_i)=0$, then we may replace T_1 and T_k maintaining their on-time status;
- $d_i < d_k$ and $R_1(T_i)=1$, then π is consistent with Algorithm 1 /cf. Step 3a of the algorithm/.

Thus, we have proved the optimality of Algorithm 1, since if there exists any optimal schedule π, using steps i, ii, iii, iv, and v, we may get, in a finite number of steps, an optimal schedule π' which is consistent with Algorithm 1. □

It is easy ti verify that Algorithm 1 takes $O(n^2)$ time.

3. NP-COMPLETE SCHEDULING PROBLEMS

In this Section we give theorems concerning the complexity of the remaining scheduling problems. Proofs may be found in [5].

Theorem 2

Problem $n|2|s=1$, forest, $\tau_i=1$, $nonpr|C_i \leq d_i$ is NP-complete.

Theorem 3

Problem $n|2|s=1$, $m_1=1$, $<$, $\tau_i=1$, $nonpr|C_i \leq d_i$ is NP-complete.

Theorem 4

Problem $n|3|s=1$, $\tau_i=1$, $nonpr|C_i \leq d_i$ is NP-complete

We give here also an interesting result concerning the polynomial reducibility among the problem of minimizing maximum lateness and the problem of scheduling to meet deadlines [5].

Theorem 5

$n|m|\lambda|L_{max} \propto n|m|\lambda|C_i \leq d_i$.

Proof

Given the problem $n|m|\lambda|L_{max}$ with deadlines d_i' and the value L of ,aximum lateness for which a "yes-no" question is answered, we can construct an instance to $n|m|\lambda|C_i \leq d_i$ in the following way: Put all the parameters the same as in $n|m|\lambda|L_{max}$, and put $d_i := d_i' + L$, $i=1,2,\ldots,n$.
It is clear that the problem $n|m|\lambda|L_{max}$ has a solution with value L if and only if the problem $n|m|\lambda|C_i \leq d_i$ has an answer "yes".

□

Following Sections 2 and 3 we may summarize the complexity of scheduling to meet deadlines in the following Table 1.

Table 1

Complexity of scheduling problems to meet deadlines

Problem complexity	m	$\{\tau_i\}$		s	m_1	References
$O(n^2)$	2	$\tau_i=1$	∅	s=1	$m_1=1$	[5]
open	2	$\tau_i=1$	∅	s=1	-	
NP-complete	2	$\tau_i=1$	forest	s=1	-	[5]
NP-complete	2	$\tau_i=1$	-	s=1	$m_1=1$	[5]
NP-complete	3	$\tau_i=1$	∅	s=1	-	[5]

4. REDUCTION TO THE NETWORK FLOW PROBLEM

As we have stated, the problem $n|2|s=k,\ r_i\neq 0,\ \tau_i=1,$ $nonpr|C_i\leq d_i$ still remains open. In this Section, we present a method for solving this problem [5], but we do not claim that it can be solved in polynomial time. Further, we will show how the problem $n|m|s=k,\ r_i\neq 0,\ \tau_i=1,\ nonpr|C_i\leq d_i$ can be solved. Namely, we reduce it to the problem of network flows with multipliers. The latter can be formulated as follows.

Let G be a directed graph with vertices $s_1, s_2, v_1, \ldots, v_p$ and arcs $e_1, e_2, \ldots, e_q$. Let $w^-(v)$ be the set of arcs directed into vertex v and $w^+(v)$, the arcs directed away from v . G will be said to denote a network with multipliers if:

a/ The source, s_1, of the network has no incoming arcs, i.e. $w^-(s_1)=\emptyset$.

b/ The sink, s_2, has no outgoing arcs, i.e. $w^+(s_2)=\emptyset$.

c/ With every vertex v_i /excluding the source and sink/ there corresponds an integer $h_i \geq 0$, called its multiplier,

d/ To each arc, e_i, there corresponds an interval $[a_i, b_i]$.

We are required to find a flow vector $\phi=(\phi_1,\phi_2\ldots,\phi_q)$ such that:

1/ $a_i \leq \phi_i \leq b_i$

2/ $h(v) \cdot \sum_{i\in w^-(v)} \phi_i = \sum_{i\in w^+(v)} \phi_i$ for all v, $v\neq s_1$, $v\neq s_2$

3/ $\sum_{i\in w^-(s_2)} \phi_i$ is maximized

ϕ_i, $i=1,2,\ldots,q$, are integers.

Solving the problem $n|2|s=k$, $r_i\neq 0$, $\tau_i=1$, $nonpr|C_i \leq d_i$ we construct a network in the following way. Each arc e_i has associated with it an interval $[0,1]$, that means either $\phi_i=0$ or $\phi_i=1$. We distinguish three groups of vertices. All $h(v)$ are equal to 1 except some vertices from the second group.

The first group represents time intervals in the shedule. For example vertex 1 represents time interval $[0, 1]$, vertex 2, time interval $[1,2]$, and so on. It is obvious that the optimal schedule must be no longer than $d_{imax}=\max_i\{d_i\}$. So the number of vertices in the first group is equal to d_{imax}.

The second group of vertices represents the possibility of processing tasks in parallel. That is, we have a vertex corresponding to the parallel processing of T_i and T_j /we denote

it by T_iT_j/ if and only if $R_l(T_i)+R_l(T_j) \leq m_l$, $l=1,2,\ldots,s$, and both $d_i > r_j$ and $d_j > r_i$. Since in this way we simulate the processing of two tasks in parallel, we have a multiplier 2 associated with each such vertex. Of course, we also have n vertices that correspond to the processing of single tasks. /Their multipliers are equal to 1/. The total number of vertices of the second group is $O\ n^2$.

We draw an arc joining vertex k $(k=1,2,\ldots,d_{imax})$ of the first group to the vertex T_iT_j (or T_i), of the second group if and only if tasks T_i and T_j (or single T_i) can be processed in the time interval $[k-1,k]$, i.e. both $r_1=\max\{r_i,r_j\} \leq k-1$ and $d_1=\min\{d_i,d_j\} \geq k$.

The third group of vertices contains n vertices which correspond to tasks. We draw an arc joining a vertex from the second group with vertex T_i from the third group if and only if this first is one of the following: T_i or T_iT_j, $j=1,2,\ldots,n$. In this way we are sure that all tasks will be processed.

It is clear that the maximal flow $\sum_{i \in w^-(s_2)} \Phi_i$ is equal to n and can be achieved if and only if all tasks meet their deadlines. The optimal schedule is constructed on the basis of the obtained arc flows.

5. FINAL REMARKS

In the class of problems of scheduling to meet deadlines under resource constraints the only problem for which an optimal and polynomial in time scheduling algorithm is known to exist is $n|2|s=1, m_1=1, \tau_i=1, \text{nonpr}|C_i \leq d_i$. The more complicated problem with an arbitrary number of resource types still remains open. Other problems have been proved to be NP-complete.

REFERENCES

1. Aho,A.V., Hopcroft,J.E., Ullman,D.D.: The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974
2. Błazewicz,J.; Scheduling dependent tasks with different arrival times to meet deadlines, in E.Gelenbe, H.Beilner /eds./ Modelling and Performance Evaluation of Computer Systems, North Holland, Amsterdam, 1976, 1976, pp.57-65.
3. Błazewicz,J.: Deadline scheduling of tasks- a survey, Foundations of Control Engineering 1, No4, 1976, pp. 2o3-216.
4. Błazewicz,J.: Mean flow time scheduling under resource constraints, Report No PR-19/77, Institute of Control Engineering, Technical University of Poznan, 1977
5. Błazewicz,J.: Scheduling with deadlines and resource constraints, Report NoPr-25/77, Institute of Control Engineering, Technical University of Poznan, 1977
6. Błazewicz,J.: Simple algorithms for multiprocessor scheduling to meet deadlines, Information Processing Letters 6, No.5, 1977

7. Coffman,E.G.,jr./ed/: Computer and Job/Shop Scheduling Theory. J. Wiley and Sons, New York, 1976.

8. Dhall,S.K., Liu,C.L.: On real-time scheduling problem /to appear/

9. Garey,M.R.,Graham,R.L.: Bounds for multiprocessor scheduling with resource constraints, SIAM, J.on Computing 4, 1975.

lo. Garey,M.R.,Johnson,D.S.: Complexity results for multiprocessor scheduling under resource contraints, SIAM J.on Computing 4, 1975, pp. 397-411.

11. Karp,R.M.: Reducibility among combinatorial problems, in R. Miller and J.Thatcher /eds./ Complexity of Comp. Computat 1972.

12. Krause,K.L.,Shen,V.Y.,Schwetman,H.D.: Analysis of several task - scheduling algorithm for a model of multiprogramming computer systems, J.ACM. 22, 1975, pp. 522-55o.

13. Labetoulle, J.: Some theorems on real-time scheduling, in E.Gelenbe, R.Mahl /eds./ Computer Architecture and Networks, 1974.

14. Labetoulle,J.: Real time scheduling in a multiprocessor environment, /to appear/

15. Lenstra,J.K.,Rinnooy Kan, A.H.G.,Brucker,P.: Complexity of machine scheduling problems, Ann.Discrete Math. /to appear/

16. Liu,C.L.,Layland,J.W.: Scheduling algorithms for multiprogramming in a hard real-time environment, J.ACM 2o,No.1. 1973. pp. 46-61.

THE STATISTICAL BEHAVIOR OF A CHARACTERISTIC OF THE INSERTION ALGORITHM "BATCHED HASHING AND LINEAR PROBING"

A. Krámli J. Pergel

In this note we investigate the stochastic behavior of some statistics arising from the insertion algorithm "batched hashing and linear probing" /see [1]/. In the literature /see e.g. [2]/ there are results for the probability distribution of the most important characteristics of such algorithms, but their time evolution - from the point of view of common distributions - is less examined. In [3] there is proved the Markovity of the process of the number of connected full intervals in the case of "hashing an linear probing", moreover the transition probabilities are determined. Now we generalize these results to the case of batched hashing.

First we give a short description of the algorithm. Let a file consist of N blocks /buckets/ and each block can contain k records. If a new record r_t is to be inserted, then a hash function $h(r_t) \in \{1,\dots,N\}$ assigns a block address to r_t. If the $h(t_t)$-th block contanins less than k elements /k is a fixed natural number/, then the new record is placed into this block. If it contains k records, then the record will be placed in the first block of the sequence $h(r_t), h(r_t)+1, \dots$ Containing less then k records. /The sign "+" is understood mod N, which means that the file is considered to be circular/. The sequence $\{h(r_t)\}$

$(t=1,2,\dots,)$ is supposed to be a sequence of independent identically distributed random variables. Let us introduce some notations. The vector $\bar{\Theta}(t) = (\Theta_1(t), \dots, \Theta_N(t))$ describes the state of the file after the insertion of the t-th record; $\Theta_i(t)$ is the number of records in the block i. The vector $\bar{\xi}(t) = (\xi_0(t), \dots \xi_k(t))$ is the statistics /a function of $\bar{\Theta}(t)$/ the stochastic behavior of which is to be investigated.

(1) $$\xi_i(t) \begin{cases} = \text{the number of blocks containing i records} \\ \quad \text{if} \quad i < k \\ = \text{the number of connected intervals of} \\ \quad \text{blocks containing k records if} \quad i = k \end{cases}$$

Notice that $\xi_0(t)$ is determined by $t, \xi_1(t), \dots, \xi_k(t)$ but for the sake of the simplicity of the treatment we assume it as a component of $\bar{\xi}(t)$.

<u>Theorem 1.</u> The vector process $\bar{\xi}(t)$ is Markov.

<u>Proof.</u> First we give the possible types of transitions

$$\bar{\xi}(t) \longrightarrow \bar{\xi}(t+1)$$

I. There is a unique i $(0<i<k)$ such that $\xi_{i-1}(t+1) = \xi_{i-1}(t) - 1$

$$\xi_i(t+1) = \xi_i(t) + 1 \qquad \text{and} \qquad \xi_j(t+1) = \xi_j(t)$$

for every $j \neq i-1, i$

II. $\xi_k(t+1) = \xi_j(t)$, $\xi_{k-1}(t+1) = \xi_{k-1}(t) - 1$ and

$$\xi_j(t+1) = \xi_j(t) \qquad \text{for every} \quad j < k-1$$

III. $\xi_k(t+1) = \xi_k(t)$ and there is a unique $i \in \{0, 1, \dots, k\}$

such that $\xi_i(t+1) = \xi_i(t)+1 \qquad \xi_{i-1}(t+1) = \xi_{i-1}(t)-1$

and $\xi_j(t+1) = \xi_j(t)$ for every $j \neq i-1, i$

IV. $\xi_k(t+1) = \xi_k(t)$ and $\xi_{k-1}(t+1) = \xi_{k-1}(t)-1$

and $\xi_j(t+1) = \xi_j(t)$ for every $j < k-1$

V. $\xi_k(t+1) = \xi_k(t)-1$ and $\xi_{k-1}(t+1) = \xi_{k-1}(t+1) - 1$

and $\xi_j(t+1) = \xi_j(t)$ for every $j < k-1$.

For the proof of the Markovity of the process $\bar{\xi}(t)$ we have to verify the relation

$$(2) \quad P\left(\bar{\xi}(t+1) \mid \bar{\xi}(t), \bar{\xi}(t-1), \dots\right) = P\left(\bar{\xi}(t+1) \mid \bar{\xi}(t)\right)$$

Relation (2) for the tranzitions of type I is a simple consequence of the symmetry of the insertion algorithm. In the remaining 4 cases the proof of (2) can be carryed out analogously /replacing formula (3) by the corresponding ones/ therefore we do it only for transitions of type V .

For a configuration $\bar{\theta}$ we denote by $\{\ell_{\bar{\theta}}^{(1)}, \ell_{\bar{\theta}}^{(2)}, \dots, \ell_{\bar{\theta}}^{(\nu_{\bar{\theta}})}\}$ the sequence of connected intervals of full blocks and by $\ell_{\bar{\theta}}$ the sum of the lengths of the connected intervals of full blocks, which are followed by a unique block containing k-1 records and a further interval of full blocks. By the definition of the insertion algorithm:

$$(3) \quad P\left(\xi_k(t+1) \mid \xi_k(t)+1, \bar{\theta}(t)\right) = \frac{\ell_{\bar{\theta}}}{N}$$

Let us denote by $\mathcal{A}^{l_1,\dots,l_\nu}(t)$ the set of all configuration $\bar{\theta}(t)$ for which the sequence $\{ l^{(1)}_{\bar{\theta}(t)},\dots, l^{(\nu_{\bar{\theta}(t)})}_{\bar{\theta}(t)} \}$ is a permutation of the natural numbers $l_1,\dots,l_\nu$.

By the formula (3) and the symmetry of the insertion algorithm at every moment t under the permutations of the addresses preserving the set of natural numbers

$\{ l^{(1)}_{\bar{\theta}(t)},\dots, l^{(\nu_{\bar{\theta}(t)})}_{\bar{\theta}(t)} \}$ we get the properties:

(4) $P(l^{(1)}_{\bar{\theta}(t)},\dots, l^{(\nu_{\bar{\theta}(t)})}_{\bar{\theta}(t)})$ does not depend on the order of numbers $l^{(1)}_{\bar{\theta}(t)},\dots, l^{(\nu_{\theta(t)})}$,

and

(5) $P(\xi_k(t+1) = \xi_k(t)+1 \mid \mathcal{A}^{l_1,\dots,l_\nu}(t))$ depends only on the state $\bar{\xi}(t)$.

The property (4) remains valid under every condition on the past $\bar{\xi}(t-1), \xi(t-2),\dots$, which together with property (5) proves (2) for transitions of type V.

A b s t r a c t

In this talk the authors prove that a statistics of the record insertion algorithm "batched hashing and linear probing" forms a Markov process.

References

[1] Blake, I.F. - Konheim, A.G., Big Buckets Are /Are Not/ Better!, JACM, 24 /1977/ 4. 591-607.

[2] Knuth, E.E., The Art of Computer Programming, Vol. 3, Addison-Wesley, Reading-Menlo Park-London-Don Mills 1973.

[3] Krámli, A., Pergel, J., Approximation of a record insertion algorithm by random walk process, Problems of Control and Information Theory 6 /4 207-211.

András Krámli
Computer and Automation Institute
Hungarian Academy of Sciences

József Pergel
Coordination and Scientific Secretariat, Computing Center of the Hungarian Planning Office

A STOCHASTIC MODEL IN PRESENCE OF INTERMITTANT FAILURE

Benczúr A., Krámli A.,

In paper [8] we published our first results concerning a stochastic model for the behaviour of a computer system in presence of intermittant failures. We have deduced our model from logical considerations of works [1] and [2] assuming that the failures can be localized to the loss of the content of central memory /quick recovery/. Led by practical experiences first we were interested in time consuming update processes which are typical for batch systems.

The cost of an update run is a convex /nonlinear/ function of the extra time caused by failures, therefore we had to investigate not only the mean value of this extra time, but the asymptotic behaviour of its probability distribution.

Our model is very close to models given by Chandy [3] and Gelenbe [4]. For the simplicity of treatment we assume the time to be discrete, and we neglect the time of creating a check point /which is practically equal to a constant value/.

Further we assume that the recovery time is equal to the time Y_t of the normal working of the system after the last check point / Y_t means the age of the server in [4] /, if during the recovery no failure occurs. In the opposite case, when a failure occurs, then the system returns again to the last checkpoint, e.t.c. - so in our model failures are allowed during the recovery procedure.

We denote by $\{\tau_k\}$ the sequence of time intervals between two consecutive failures, which forms a sequence of i.i.d. random variables, i.e. the failure process is a general renewal process. The sequence $\{\zeta_k\}$ of time intervals between two consecutive checkpoints measured by the time of the normal working of the system is a sequence of i.i.d. and bounded random variables with the common distribution function $F(y)$. We assume that the processes $\{\tau_k\}$ and $\{\zeta_k\}$ are totally independent.

If the failure process is Markov renewal /i.e. τ_k is geometrically distributed with the parameter p /, then we can replace the cost $h(Y_t)$ of the recovery introduced in [4] by the expected time $\bar{h}(Y_t)$ of the recovery procedure disturbed by random failures (Y_t is the age of the server).

So, from the point of view of expected extra time caused by failures our model can be reduced to Gelenbe's one.

But if we are interested in the probability distribution of this extra time, or the failure process is not Markov renewal, then our model is more adequate.

It is easy to calculate that under the above assumptions $\bar{h}(Y_t) > C\, p^{-Y_t}$ see [5]. So, by [4] the stationary probability of the normal working of the system is equal to 0, if the momentum generator function $\sum_{j=1}^{\infty} e^{j\varepsilon(F(j)-F(j-1))}$ of the distribution function $F(y)$ does not exist for $\varepsilon \leq \ln \frac{1}{p}$. This fact makes natural our condition on the boundedness of ζ_k.

Let us introduce the random processes $\{\theta_k\}$ and $\{\varepsilon_k\}$

$$\theta_1 = \max_{\sum_{j=1}^{\ell} \gamma_j < \tau_1} \sum_{j=1}^{\ell} \gamma_j$$

$$\vdots$$

$$\theta_k = \max_{\sum_{j=1}^{\ell} \gamma_j < \sum_{j=1}^{k-1} \theta_j + \tau_k} \left(\sum_{j=1}^{\ell} \gamma_j - \sum_{j=1}^{k-1} \theta_j \right)$$

$$\varepsilon_k = \tau_k - \theta_k$$

The random variable θ_k is equal to the distance between those checkpoints from where the $k-1$-st and the k-th recoveries were made, and ε_k is the extra time caused by the k-th failure.

Set $\nu_n = \min_{\sum_{j=1}^{\ell} \theta_j = \sum_{j=1}^{n} \gamma_j} \ell$ (ν_n is the number of failures occuring up to the n-th checkpoint.)

If $P(\gamma_k = d) = 1$, then $\{\varepsilon_k\}$ and $\{\theta_k\}$ are sequences of i.i.d. random variables, and Anscombe's central limit theorem can be applied to the normed sums

$$\sum_{k=1}^{\nu_n} \frac{\varepsilon_k - E\varepsilon_k}{\sqrt{\nu_n}}$$

If $P(\gamma_k = \ell) \neq 0$ for every $\ell \leqq \max \gamma_k$, then the sequence $\{\varepsilon_k\}$ can be embedded into an irreducible aperiodic Markov chain $\{\bar{\varepsilon}_k\}$ with finite state space, therefore there exists a unique stationary distribution for $\{\bar{\varepsilon}_k\}$. It can be shown that if the process $\{\bar{\varepsilon}_k\}$ is stationary, than the process $\{\theta_k\}$ is also stationary. Set $E\varepsilon_k = m$, $E\theta_k = c$, where the expectation is taken on the basis of stationary distributions. The stationary sequence $\{\varepsilon_k\}$ satisfies the

strong mixing condition, and Rozanov's condition see [7] of the validity of the central limit theorem. Under further conditions on processes $\{\tau_k\}$ and $\{\gamma_k\}$ we can prove the central limit theorem for the sums

$$\sum_{k=1}^{\nu_n} \varepsilon_k$$ too:

THEOREM 1. If $\{\tau_k\}$ has finite 8 - th moment and $E\left(\sum_{k=1}^{n}(\varepsilon_k - m)\right)^2 \to \infty$, then there exists a positive number σ such that

$$\lim_{n\to\infty} P\left(\frac{\sum_{k=1}^{\nu_n}(\varepsilon_k - m)}{\sigma\sqrt{\nu_n}} > x \right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{u^2}{2}}$$

We can formulate the queueing problems analysed by Gelenbe in our recovery model too.

We assume that the queue can be arbitrarily long, the probability of the arrival of a request at the moment t is equal to $p < 1$ independently of the state of the system and the service time - measured by the time of the normal working of the system - is geometrically distributed. Let ξ_t be the length of the queue at t.

The process $\{\xi_t\}$ can be regarded as a doubly stochastic Markov chain the transition probabilities of which depend on the realizations of processes $\{\tau_k\}$ and $\{\gamma_k\}$ /here τ_k does not contain the time, when the queue is empty/. The process $\{\xi_k\}$ can be embedded into a Markov chain $\{\bar{\xi}_k\}$ with countable state space. Using the elementary methods of renewal theory there can be proved the following:

THEOREM 2. The process $\{\bar{\xi}_k\}$ has stationary distribution if and only if $pm < (q-p)c$.

Abstract

This talk gives an example for the application of central limit theorems in the calculation of the probability distribution function of the extra-time consumed by recovery procedures ensuring the reliable working of DBMS.

References

[1] CODASYL Comittee Data Base Task Group Report, Association for Computing Machinery, April, 1971.

[2] FOSSUM, B.M., Data base integrity as provided by a particular data base management system, Proceedings of IFIP Working Conference on Data Base Management, Cargese Corsica, France, 1-5 April, 1974.

[3] Chandy, K.M., Browne, I.C., Dissly, W.R. Uhring - "Analytical models for Roll back and Recovery Strategies in Data Base Systems" - IEEE Transactions on Software Engineering - Vol. 1, n° 1, - pp.loo-llo - March 1975.

[4] Gelenbe, E. On the optimum checkpoint interval /Manuscrit/

[5] Vold, H., Sjørgen,B.H., Optimal Backup of Data Bases: A Statistical Investigations, BIT, 13, 1973 pp. 233-241

[6] Rényi, A., On the central limit theorem for the sum of a random number of independent random variables, Acta Math. Acad, Sci. Hungar.v. 11, 196o.pp.97-1o2.

[7] Ибрагимов, И.А., Линник, Ю.В., Независимые и стационарно связанные величины, НАУКА, Москва, 1965.

[8] Benczúr, A., Krámli, A., A note on data base integrity Acta Cybernetica, Tom. 3, Fasc. 3, Szeged, 1977. pp.181-185

OPERATING SYSTEM AND DATA BASE FOR A SMALL PRODUCTION CONTROL SYSTEM

A.Wolisz, A.Krawet, J.Mierzwa, A.Nowakowski

1.INTRODUCTION

Authors of this paper took part in the projeet of developing a dispatcher-aiding computer system for the soaking pits division of iron and steelworks. By now, the system has been succesfully operating for 6 months, and considerable amount of experience on it's exploitation has been obtained. The problem of system's functional characteristic, and scope of application software for such a plant will be discussed in a separate paper, being mentioned here briefly to give some impression of the problem whitch was solved. On the other hand, some solutions introduced in the structure of operating system, application software and data base organisation are believed to be of more general type-pertaining possibly to a larger class of discrete type production control systems.

2.PLANT DESCRIPTION

The soaking pits division is a crucial part of the so called "hot steel line" in a steelwork, situated between the melting shop and slabbing mill. The division should provide the slabbing mill with a constant stream of properly heated ingots, independently of possible perturbances in melting shop operation. In order to do so, it is provided with a local stock of ingots /a local buffer/ and cooperates closely with a cold ingots magazine.
The scheme of material flow in soaking pits division is presented in fig.1.
In the considered case the soaking pits division consisted of 40 pits, each of them allowing for the average load of 100 tons and being recharged in average once during a shift.

Ingots were delivered in batches corresponding to consecutive casts, consisting of 300-350 tons of steel. Up to 3000 tons of steel could be hold on the local stock. Up to twenty mould types and 150 types of steel were to be considered.

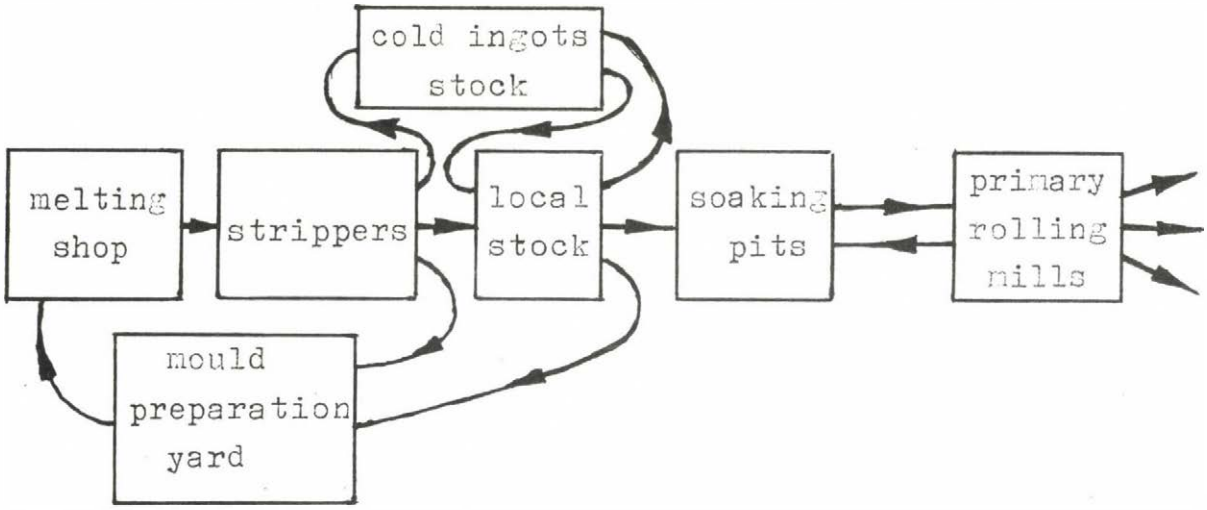


fig.1

3.SYSTEM FUNCTIONS

The computer system was assumed to operate in a real-time off-line mode, being used by a dispatcher in a conversational way. The main functions were following ones :

a/ keeping track of the material flow through the plant, including :
   - perserving information about the load of every soaking pit
   - perserving information about the local stock of ingots
   - constant displaying of the pit's actual state

b/ suggesting the way of heating and time to be spend in the pits, including :
   - classification of ingots to various groups
   - charging temperature calculations
   - determining the number of process phases and their duration

c/ heating monitoring, consisting in displaying the time-to-completion for every phase and alarming the dispatcher if any process phase has not been completed in proper time

d/ reporting - including shift reports and current reports

All the information about the division's operation is introduced by the dispatcher through an alfanumeric keyboard, in conversational units corresponding to elementary technological operations.

4. SYSTEM CONFIGURATION

The scheme of the utilized computer system configuration is presented in fig.2.

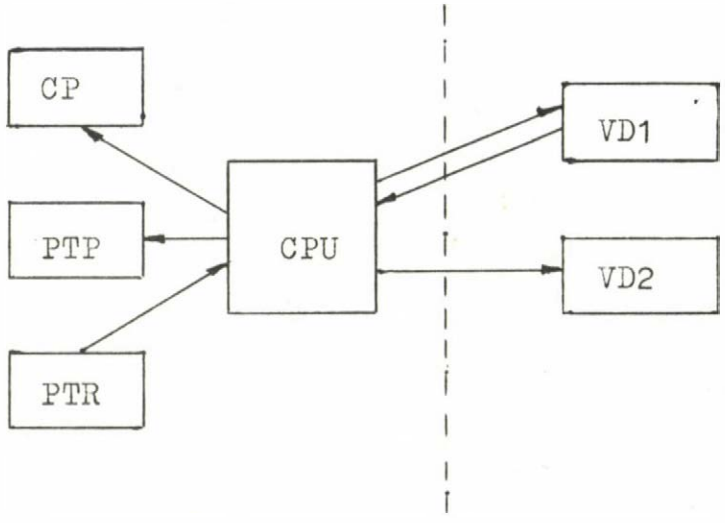


fig.2

The equipment consist of :

- Central Processing Unit with 16 kword core memory /CPU/
- 2 Video-Displays /VD1 and VD2/ with alphanumeric Keyboards
- Character Printer /CP/
- Paper Tape Reader /PTR/
- Paper Tape Punch /PTP/

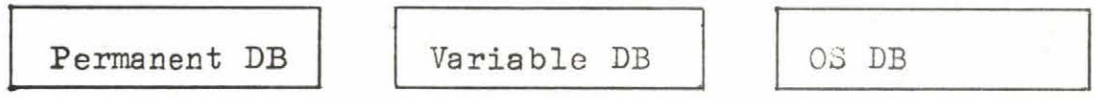
Peripherals were utilised in the following way :

- The PTR was used for introducing the code and data exclusively
- The PTP was used for obtaining the history of every heating

cycle, to be later used as a source information for accountancy, technology analysis and quality inspection
- The CP was used for reports printing and obtaining hard copy of some actions
- The VD1 was used for conversation - it is data introducing and obtaining information about the suggested decisions
- The VD2 was used for displaying the state of soaking pits division

5. GENERAL SOFTWARE ORGANISATION

We shall now discuss briefly the software and data base organisation, presented in fig.3 following representation introduced in /1/.



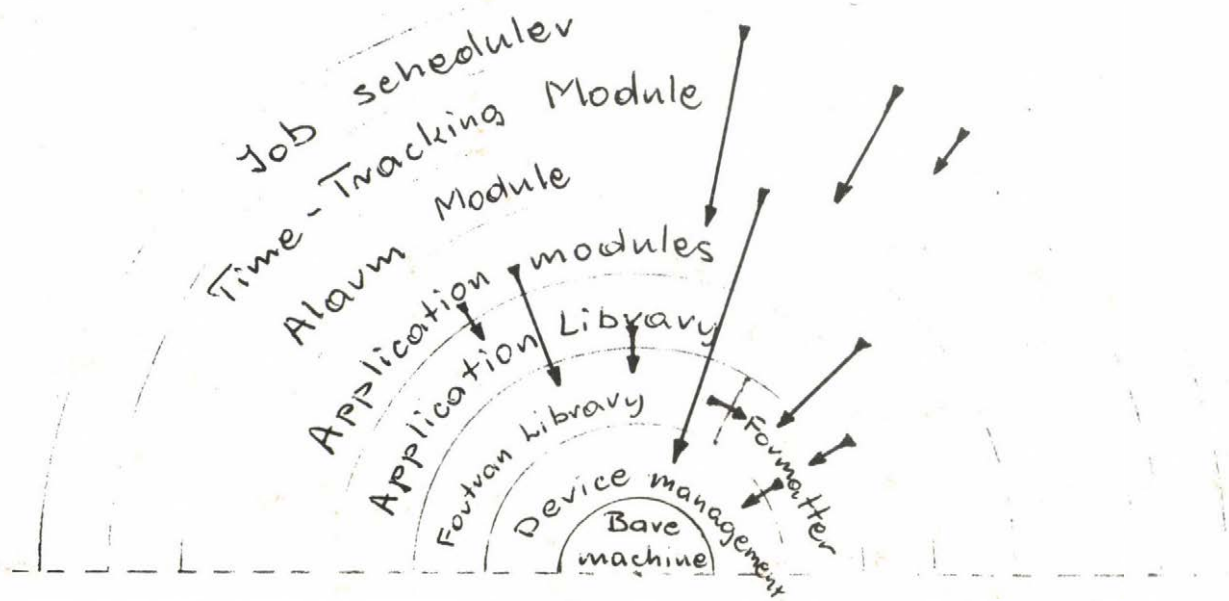


fig.3

A detailed discussion of some main parts will follow in consecutive sections. The data base is devided into three main parts :

a/ The permanent data base /PDB/
PDB contains information about the mathematical models included in the system, eg. the cooling model, the heating model. The PDB may be changed exclusively through reading from the paper tape under the operating systems initial start module supervision. It is of the read-only type for all programs included in the system.

b/ The variable data base /VDB/
VDB contains information about the actual state of the plant, thus enabling monitoring of the material flow and soaking pits operation.

c/ The Operating System data base /OSDB/
OSDB contains all information desired for program scheduling, buffers for inter-program communication, and device management tables.

Program modules included into the system have been devided into several levels /fig.3/ grouping those having equal access rights to other modules and various parts of data base. Arrows represent the privilege of calling lower levels by modules occupying higher levels.

a/ The lowest level consisting of I/O traffic control routine and drivers for peripheral devices, handles the I/O requests. All the peripherals except of video-displays operate in a fully asynchronous way, with speed controled by the CPU. The interface standard is in agreement with the one utilised in Hewlett-Packard computers /2/. Interfacing of the video-display /Videoton VT 340/o-type/ is different because in the SEND mode and ACTIVISATION LOOP their operation speed is computer independent and information loss has to be avoided. Detailed discussion of this problem is given in the Appendix.

b/ The FORTRAN LIBRARY contains a set of routines for standard mathematical and logical operations. It includes also a special program, the FORMATTER, used as the single interfa-

ce between I/O requests of the applications programs and the device management level. FORMATTER allows for data input and output in the desired, predefined format.

c/ The Application Library /ALIB/
ALIB contains a set of routines prepared to operate on VDB. They perform data storage and retrieval in specific data structures used in VDB, like packing and unpacking several data items into one memory location. They perform a number of sorting and searching operations according to the defined key. ALIB contains also a subroutine for editing standard massages and input data control routines for data correctness checking.

d/ The Application Modules, are designed as conversational units for man-machine communication. Eighteen such modules included in the system may be requested by the dispatcher through appropriate operations codes. Those modules are closely connected with various technological situations, about which either data are to be introduced, or some information is to be obtained from the system. After choosing the proper module the dialog is controled by the computer, which asks consecutive questions. The dispatcher has to introduce numerical or alphanumerical data of defined type when requested. As sometimes a logical-type answer /YES or NO/ is needed so called DECISION POINTS have been defined, were the answer YES, NO or CHANGE THE MODULE >MODULE CODE< is to be given. When no application module is being executed, a pseudo module - waiting for a new module request is in progress. Both ALIB and Application Modules are written in FORTRAN.

e/ Alarm Module is utilised on operator's request, communicated thrugh the CPU switch register, in the case of peripheral devices failure, in order to arrange proper reconfiguration. This module may be also called when the device improper operation has been established by the Operating System. It has access to the device management level, to break the improper I/O operation and clear the device status.

f/ The Time-Tracking Module, is in charge of monitoring the time-dependent events, emiting proper signals if a module connected with such an event should be executed.

g/ The Job Scheduler, is in fact a sophisticated interval clock's driver. The utilised CPU does not have a real-time clock, but only an interval clock generating interrupts with frequency 1/sek. The Scheduler is being executed with such a frequency, analysing the state of the computer system, possible requests for module's execution, signals from the time-tracking module which it activates every minute and other signals edited, while application module was executed. It organises also a real-time clock in the way of interval counting. After the analysis and testing of some I/O devices conditions, a proper program is started /including the possibility of resuming the interrupted one/.

Remark : As modules f and g operate interrupting the execution of lower levels, which are not of reentrant type, neither the time-tracking module nor the scheduler may use any of the a, b, c levels modules.

The scheduler includes an initialisation section, seting system's starting conditions according to the needs represented by the switch register. There is possible to start with any combination of following actions :

- reading the code from PTR device
- reading the content of PDB from the PTR device
- clearing the content of VDB

Afterwords the initial state of VDB can be introduced in a conversational mode, and the clock is set, starting normal operation.

6. DATA BASE DESCRIPTION

In the previous section the data base has been briefly described. Now some more details on it's organisation shall be given. The PDB consists of 14 files, defined as either vectors or two dimensional matrices. In the files data are organised in most cases in a relational structure, of either simple or hierar-

chical type. Two files used as dictionaries from technological symbols into internal code are accessed by direct search of the key. This structures where chosen so as to minimise the joint memory requirements of the PDB and using it programs.

The VDB has a more general structure which can be used also in other discrete-type production control systems. It consists of three main files: the agregate file /AF/, the material-batches file /MBF/ and the report file /RF/. The AF has a constant number of fixed length records, equal to the number of agregates /soaking pits/. The file is represented as a relational data structure of the two-dimensional matrix type where every row discribes the state of a single agregate /3/. The agregate number points directly to a record. Every record contains 19 fields coded into 15 words of memory. The information is changed after every technological operation.
The MBF has a variable /with upper limit/ number of fixed length records, equal to the number of material batches /casts/ served during the current shift in the division. Like previously the file is represented as a two-dimensional matrix, each row containing information about a single cast. The first field in every record contains the key /cast number/, and proper records are found by direct search. Key equal to zero denotes the end of file. Each record consists of 17 fields coded into 12 memory words. The information is changed after every technological operation and after the shift end the completly finished casts are removed.
The RF file is a vector structure containing information about the production progress. The data are modified after every operation connected with the material flow.
The OSDB is divided into three main parts:
a/ the inter-program communication area is accessible for upper software levels /d - g/,it contains :

- Logical Device Table enabling adressing of the logical devices, and peripherals interchanging by software means in the case of failures
- Information about the actual time and data

- Flags signalling conditions for some application program according to the situation in which they are called, in order to modify their functions according to Operating System's decision.

b/ Scheduler's data base available for this program exclusively consists of following informations :

- buffer for human operator commands
- stack for preempted programs
- pointer to the actually executed program
- program table containing information about application programs supervised by the scheduler /fig.4/. Priority of the programs is given by their order

the alarm routine

activation index	starting adress

periodical program

activation index	starting adress	counter	period

application modules /operator driven/

activation index	starting adress	operator code

activation index is set for every program which has been found by the scheduler to meet conditions for execution /for detailes look in section 7/

Program table entries formats.
fig.4

c/ Device management tables, containing the actual information about the peripheral devices included into the system, and information about the actualy fulfilled operation. Those being a part of standard software obey, like the I/O routines the HP standard. This part contains also special VD-driver's flags described in the Appendix.

7. PROGRAM STATES

The graph describing possible program's states is given in fig.5.

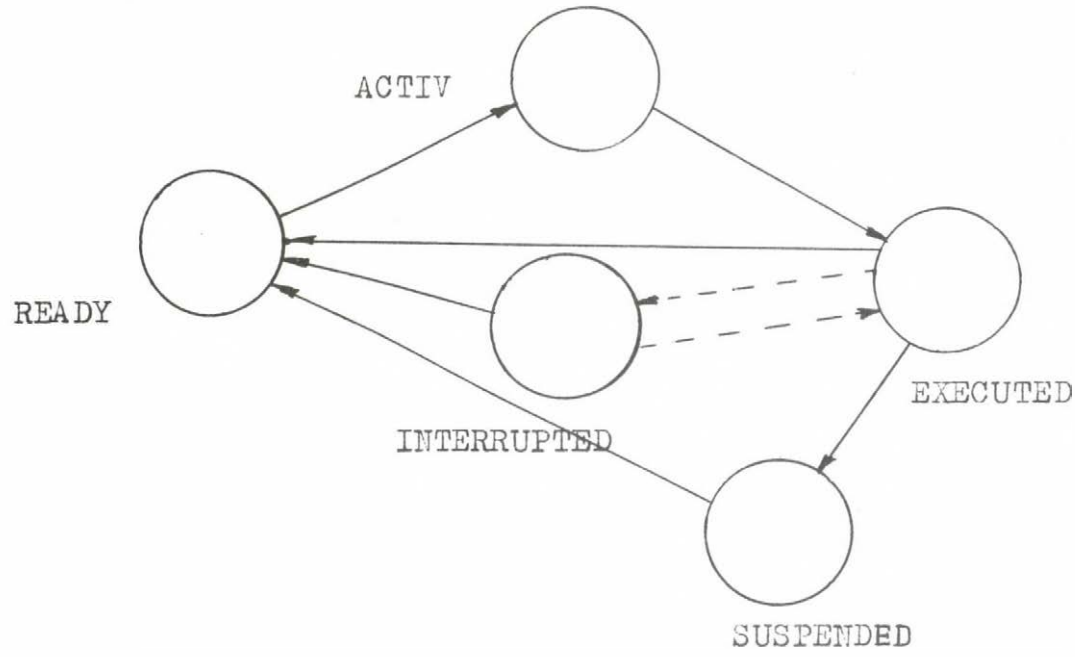


fig.5

The usual program's state is READY, which can be changed to ACTIVE if proper conditions /eg. time period, internal conditions switch register change or operator request/ have been satisfied. This transition is fulfilled by changing activation index of the proper program to one /fig.4/ by the scheduler, or in case of the periodic programs, by the time-tracking module, being in turn activated every full minute by the scheduler. From this state on the priority basis defined by the position in program table only one program is chosen and moved to the state EXECUTED, which is marked in the actually executed program pointer. The time-tracking module and alarm routine are moved to this state according to the preemptive priority rule, while other programs are scheduled according to the head of the line discipline. While in the EXECUTED state the program is interrupted allowing for scheduler's and time-tracking module's activity.

thus being temporarily move to an INTERRUPTED state. If during the scheduler's activity a transmission fault is detected the interrupted program is not resumed, and it's status is changed for READY. After program completion the state is changed for READY. If during execution of a conversational type program operator decides in a DECISION POINT to change it, using a proper code /that code is read to a buffer/, the program is moved to a SUSPENDED state, and during the nearest scheduler's operation moved to the READY state, while another program is chosen for execution on the usual basis. Thus the response time for operator request has an upper equal to 1 second.

8. RELIABILITY PRECAUTIONS

We shall now describe shortly precautions which have been taken to assure system's reliable operation. A carefull analysis of possible operating errors has been fulfilled on the design stage, resulting in software structure. It has been assumed basing on the laboratory experience that the CPU will be highly reliable /in fact two CPU failures occured during 8 months, including one caused by external factors/, the peripheral devices being more vulnerable. The human operator was expected to be the most dangerous part of the system, being suspected to undertake-conciously or by mistake a lot of strange and misleading actions. Basing on those assumptions following security steps have been incorporated :

a/ The proper introducing of code and data has been assured by
   - reading the code in records with checksums
   - reading data /the PDB/ in the form of 30 records, containing 37 ASCII characters each. The parity is checked for every character, while records are provided with checksums. /the early version constrained only to parity control was found unsufficient/. In the case of code or PDB corruption /eg. as a result of very hard electromagnetic disturbances/ it is possible to introduce it again, perserving the other parts of data base. In fact such event was observed a few times.

b/ Precautions against peripheral failures included :
- a specific, highly reliable communication mode with the video-displays /look Appendix/, including testing of the transmission end emergency procedures.
- the possibility of destroing any I/O operations with device clearing, if desired, allowing to ask for that action from the CPU swith register in the normal operation mode.
- the possibility of reconfiguration of peripheral devices if desired, allowing to control this action from the CPU swith register. In fact it is possible to continue operation only with one video-display, which is selected by the operator. Reports can be produced on the paper tape punch in the case of printer's failure.

c/ The operations code format is defined as :

In the basic state /it is waiting for requests/ all other messages are ignored.

d/ Dialog mode was defined so that after recognising the proper operations code every data is asked for with detailed questions /eg. Cast Number - ..., Mould type - .../.
In a normal mode the video-display keybord is inhibited and in order to activate it the TYPE key has to be pressed. After typing the data a SEND key has to be pressed in order to enable the transmission. This feature of the used VD prevents accidental information generation. Every information introduced in an application module is stored during its execution in working buffer, thus enabling the operator to break dialog and distroy the data if, basing on some computer answers he finds them wrong. Only after completing the whole module, data are stored in the VDB.

e/ Every introduced piece of data is tested
- visually, by the operator between typing and transmiting
- by the special input routine, to check if the data is of the desired type, the possibilities being - numeric or

alphanumeric. Afterwords numeric data are compared with the upper and lower value for the specific data type /the system allows for 30 pairs of boundary values/. Improper data are dropped, and the question for data is repeated.

- if the introduced data describes on which agregate /soaking pit/ or material batch /cast/ the technological operation defined by the operation's code was fulfilled, it is checked if this operation is permitted on this very object, basing on the possible operations sequences. Illegal attempts are refused. Example: Unloading ingots in the heating phase is prohibited.
- if the introduced data stays for the quantity, it is tested if the defined operation on such a quantity is possible, and if such a quantity is available in the very situation. Example: While attempting to charge 10 ingots from this cast remain uncharged.
- in some cases outputs of special modules are also tested for upper and lower limit, thus discovering illegal combinations of data, each of which satisfies the boundary tests. In this case the temporary data buffer may be destroied and application module repeted.

f/ In the case when an improper, although reasonable data has been introduced, operator may change it easily using a special module enabling direct acces to the AF and MBF files of the variable data base. However a hard copy of every such actions is perserved. Although an attempt was done to minimize the data volume memorized in the data base in some cases that has been allowed for a bit of redundancy between those files, in order to find possible discrepences in the introduced changes pointing out an error.

9.CONCLUSIONS

The described system has been succesfully implemented in the industrial environment. Its operation in a hard environment, while served by operators withou computer science back-

ground /only ofter short training/ verified the validity of applied solutions. Especially the reliability precautions proved to be extremly usefull and efficient. It proved to be impossible to brake the system down by any improper operation. Naturally, due to the memory limitations, the functional scope of the Operating System is highly limited, creating numerous difficulties for example in introducing any changes in the system. However, the authors believe, that after slight modification and development a similar solution could fairly well cope with the needs of simple dispatcher-aiding systems.

10.ACKNOWLEDGEMENTS

The authors wish to thank Mr Romuald Pozowski, M.Sc., for a lot of valuable discussions and remarks during the system's development, which helped them in finding and implementing more efficient solutions.

11.REFERENCES

/1/ S.E.Madnick, J.J.Donovan "Operating Systems"
McGraw-Hill, New York 1974

/2/ "A Pocket-Guide to HP Computers"
Hewlett-Packard Company, Cupertino 1973

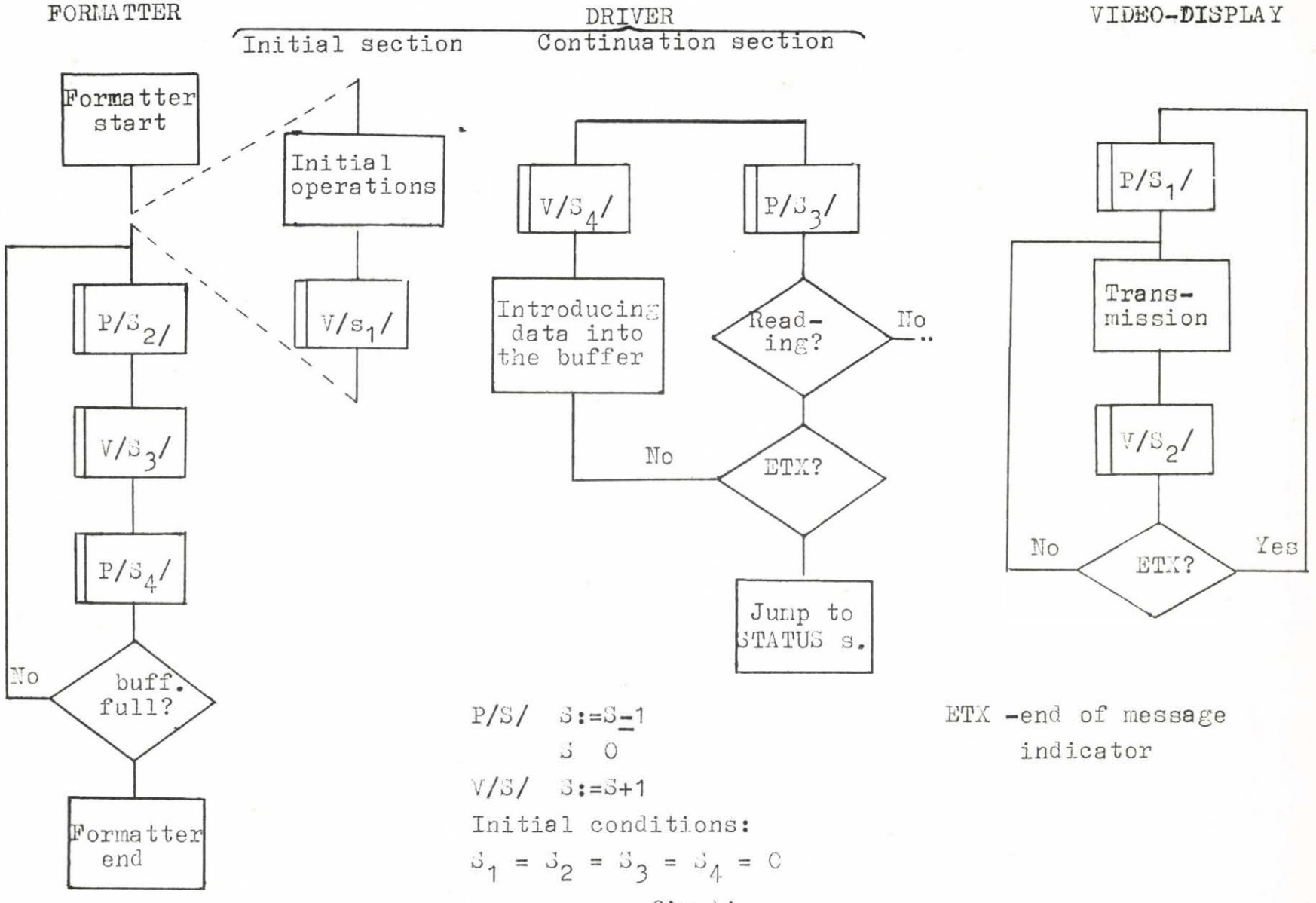
/3/ C.I.Data "An introduction to Data Base Systems"
Addison Wesley, Reading 1976

Adam WOLISZ,Ph.D,
Adam KRAWET,MSc,
Janusz MIERZWA,MSc,
Andrzej NOWAKOWSKI,MSc,
Polish Academy of Sciences,Dept. of Complex Automation,
44-100 Gliwice,Baltycka 5,Poland

APPENDIX

COMMUNICATION BETWEEN THE CPU AND VIDEO-DISPLAY

In this section we shall discuss briefly the problem of information exchange between the CPU and VD, being of a specific nature due to the VT 340/o type displays organisation. As it has already been mentioned in section the communication rules incorporated in the system are in agreement with the Hewlett-Packard standard, of the following type :
The peripheral device is activated by sending a control signal. After generating a single byte /or word/ of information the peripheral device sends the "flag" signal, which causes the CPU to read the data. If a further information unit is expected, then a consecutive control signal is given. The output process takes part in much the similar way. Thus the peripheral device works with a speed controlled fully by the CPU. It is quite different with the VT 340/o video-displays, as far as data input is concerned.
First of all, in the conversational mode of operation. the human operator has to be given the discretion when to transmit data to the CPU, this being done by pressing the SEND button on the keybord, thus causing the VD to switch into the send mode. However this does not cause sending any information to the computer. The CPU has to ask for the VD status by sending the VD's number, and only after that a status section containing three bytes of information /the status and cursor's row and column/ is transmitted. However those three bytes are being send independently from any CPU signals, with a speed depending on the VD output module exclusively! Neither is any acknowledgement signal expected. If the status section has been received the CPU may send a "transmit" command, after which the transmission starts, and goes on up the same rules as the status section transmission, that means independently from any CPU signals. The organisation of intercooperation of processes involved in the data input activity is shown in fig.A1.



P/S/ S:=S−1

V/S/ S:=S+1

Initial conditions:

$S_1 = S_2 = S_3 = S_4 = 0$

ETX -end of message indicator

fig.A1

Three processes are involved in this action, namely :

- the Formatter routine, organising the I/O operation if utility programs are written in FORTRAN
- the driver routine, organising the elementary input operation
- the VD logic, generating the data if either the status section or data block is transmitted.

The usual binery semaphore nototion is being used.

Meanings of the utilised semaphores are following :

S_1 - is the transmissinn start semaphore

S_2 - is the byte-ready for input signal

S_3 - is a driver activation signal, when the data readiness is found by the Formatter

S_4 - is a CPU ready for next transmission signal.

Analysing the scheme given in fig.A1 it is easy to prove the incorrectness of this synchronisation scheme, as the monitor's transmission loop is not influenced by the speed of other processes. Thus data loss might occur, if the reading process would not be fast enough, for exemple in the case of higher priority interrupt service. How could this be avoided ?

A trivial solution could be immediately given : one should do all the reading in an interrupt-diseabled mode. However this solution is a wrong one from the reliability reasons.

If the VD was disconnected during the transmission /it means set to the local mode by the human operator/ all the computing process would be blocked. Thus the clock has to be given higher priority and allowed to interrupt. On the other hand the clock service routine, that means the job scheduler module needs for its executinn such an amount of time, that information losses would certainly occur. The following solution has been applied :

If the status section is transmitted, a special semaphore is decremented, preventing the clock interrupt to be served, delaying it's execution until the section is finished.

In the case if the data transmission takes place, only a short part of the job scheduler is executed, delaying the rest until the first interrupt after the transmission is completed.

The variety of applied solutions was caused by the different type of both cases.
The data transmissions are comparatively rare, thus the delay in executing of the full interrupt service is permitted.
In the case of status section, it's length is very small, but the section is usually repeated constantly in a "wait loop" for the send status. Thus the clock interrupts are likely to find the status section of transmission for several consecutive times, and the full service wouldn't take place for a long time. In the utilised solution every interrupt service takes place after the section's completion, thus with a delay of a fraction of second. If the next interrupt found the previous one unserved /which is tested/ it would mean that a transmission failure occured, the transmission would be broken, and proper message for the operator would be send.
Such an organisation has proved itself correct during the system's operation.

DEADLOCK PROBLEMS IN A MULTICOMPUTER INTERCONNECTION SYSTEM

L. Simonfai
Research Institute for Applied Computer Sciences, Budapest, Hungary

Summary

An operating system called MUSCLE is being developed to supervise the operation of a multi-computer system. MUSCLE requires two hardware resources to transmit a block of data between two user processes. Several transfers are executed concurrently so the possibility of system deadlock exists. Information about the resource allocation state of the complete system is distributed among the participating computers. Allocation decisions have to be based on information locally available as opposed to conventional centralized systems.

After a brief overview of the system methods to deal with deadlock are reviewed. Two algorithms are presented both of which use only locally available information to prevent deadlock. A deadlock model due to Coffmann is used to investigate the system behaviour. The basic observations sufficient to the construction of a formal proof on the deadlock-free behaviour of the system are given for both algorithms.
A simple hardware extension is proposed to make the total allocation state available troughout the system and algorithm 3 exploits this possibility.

1. Overview of the system

A multi-computer operating system, called MUSCLE /MUlti-computer System ControL Environment/ is under development at SZAMKI.[1]. The hardware configuration is shown on fig. 1. At most 4 Rlo computers are interconnected by a single, high-speed, bidirectional bus. Each computer consists of a standard Rlo processor, memory, the usual complement of MINIBUS peripherals and a special direct memory access facility /DMA/. The DMA allows concurrent, high speed transfer of data blocks between special DMA peripherals /eg. disk, tapes etc./ and the memory, and - if connected through a bus control unit /BCU/ to the common bus - between the memory units of different processors. The actual information exchange is done autonomously using a register set /CCW/ in each BCU. This register set contains the usual channel control parameters /direction, word-count, memory address/. The necessary actions at initiation and termination of a DMA transfer are executed using the MINIBUS.

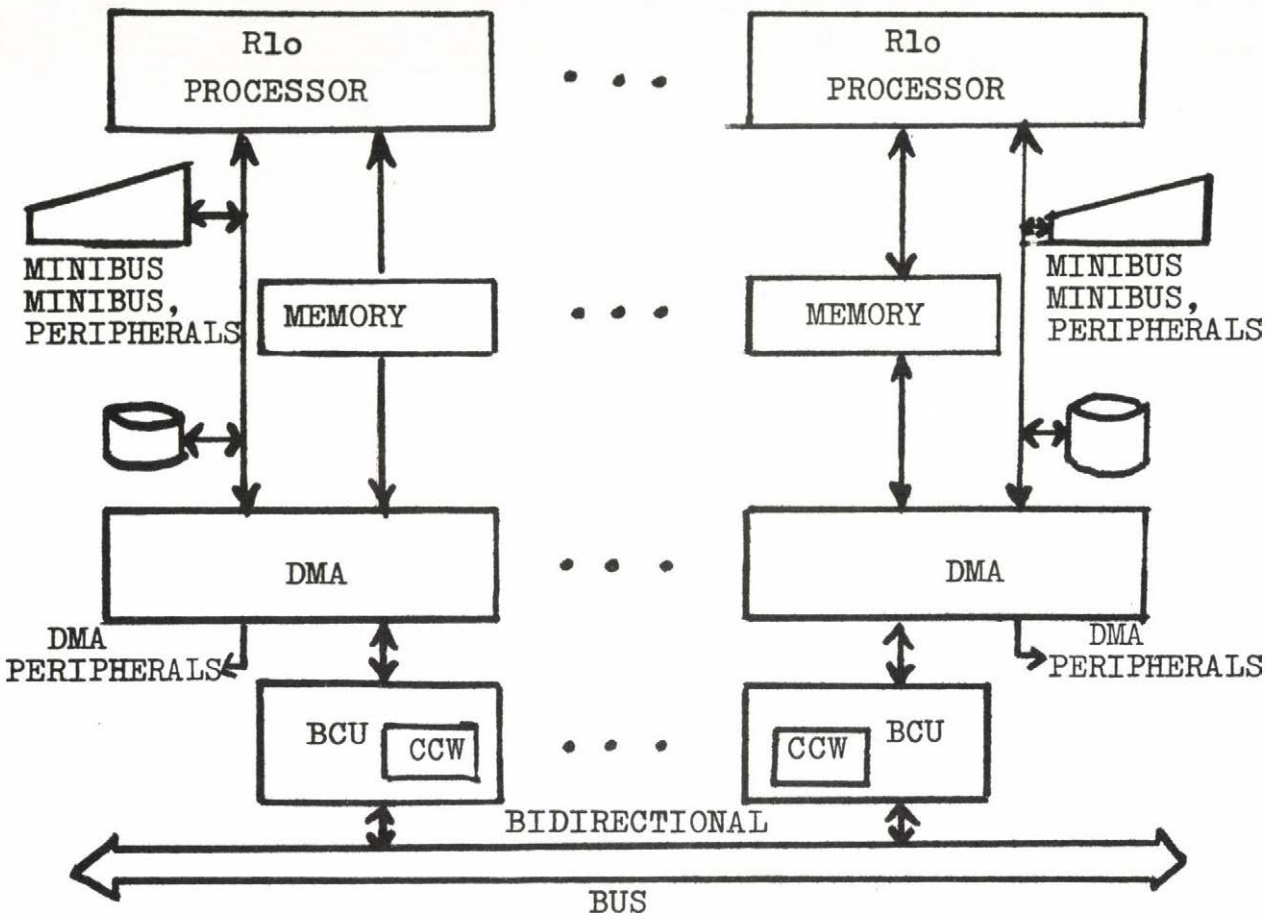
On a logical level the system consists of several user processes communicating through logic channels as shown on fig. 2. A logic channel is a named entity connecting two processes, and capable to transfer data unidirectionally between the processes. A process may possess several logic channels connecting it to the same or different processes executing on any processor in the system. Appropriate user commands are provided to open or close the logic channels and to send or receive information using them. The joint effort of the two processes issuing a send and a receive command respectively for the

same logic channel is necessary to transmit a block of data. Details of the system will be reported elsewhere.

The processors are able to send short /16 bits/ administrative messages to each other to synchronise their behaviour. The actual data transfer on a logic channel requires three consecutive actions.

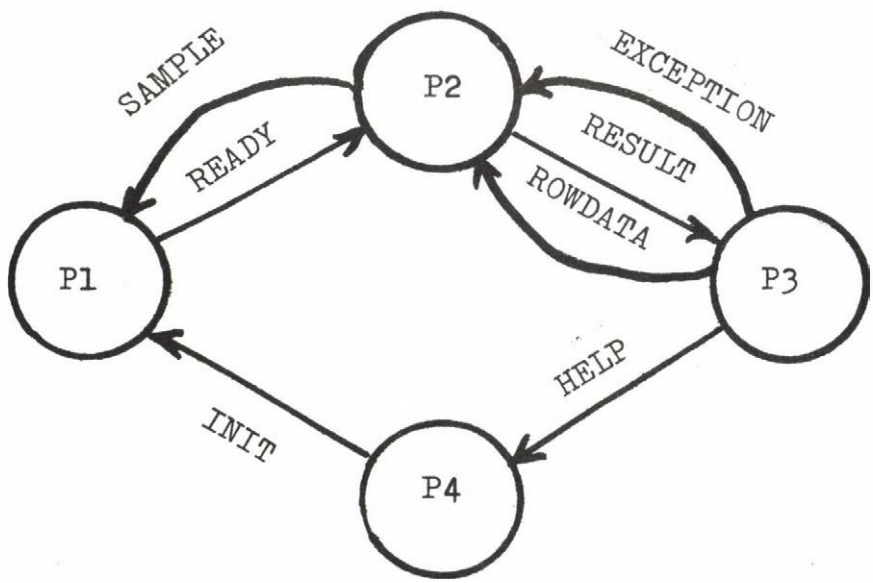
- request: signalling that the process on the request sending side is ready to transfer;
- acknowledgement: signalling that the process on the request accepting side is also ready to transfer, the CCW is loaded with the apropriate parameters and marked busy;
- transfer: after getting the acknowledgement the requestor side immediately requests its CCW, and if acquired loads it with the transfer parameters and starts the transfer, which once started runs autonomously until completion. After completion both CCWs are released. The transfer is an undivisible action whereas the bus is released after a request or an acknowledgement.

There is a complete symmetry between the processes, both of them may function as either a requestor or an acceptor depending on the actual timing conditions. Transfer on several logic channels may be in any of its execution phases simultaneously. The information necessary for the correct operation is distributed in the system. Each of the processors "knows" only a partial state of the system pertiment to its operation. To contrast all information necessary to deal with deadlock is



Hardware Configuration

Figure 1



Processes Communicating through Logic Channels

Figure 2

available in the same memory in a monocomputer system. Now the possibility of deadlock is clear. Suppose that on behalf of logic channel ROWDATA connecting user processes P2 and P3 executing on processor1 and processor2 respectively an acknowledgement was sent by processor2 so CCW in BCU2 is busy. Suppose further that on behalf of logic channel INIT connecting P1 and P4 executing again on processor1 and processor2 an acknowledgement was also sent by processor1. Now neither processor1 nor processor2 can acquire its CCW and start the transfer. A classic example of system deadlock has occured.

2. Deadlock and deadlockfree behaviour

Three conditions are necessary for a deadlock situation to exist [2]:

- mutual exclusion: each process claims exclusive control of the resources it uses;
- nonpreemption: a process does not release the resources it holds until it completes its use of them;
- resource waiting for the others to release resources.

Three approaches are possible to deal with deadlock:

- prevention: one or more of the above conditions are precluded by appropriate system design;
- detection and recovery: the joint progress of all processes in the system is monitored and recovery operations are started if deadlock is observed;
- avoidance: on the basis of advance information, resources are allocated at any instant so, that deadlock will never

occur.

The last two approaches are infeasible in our case, because
- both requires complete information about the system state;
- avoidance requires some more advance information about resource usage.

Preventation remains therefore as the only viable solution, requiring that at least one of the necessary deadlock conditions cannot hold.

Mutual exclusion of resource usage is mandatory. Preemption is possible, but has two serious disadvantages. Some kind of reset action should be provided, which complicates the basic three step transfer procedure. Moreover the bus load is increased by the additional administrative messages while it should be kept as low as possible. So resource waiting has to be regulated so as to preclude system deadlock.

3. The model

The deadlock model of Coffman et al.[2], [3] is used to formalize the behaviour of the system. The allocation graph Qk is a directed graph defined as a set of vertices corresponding to the resource types R1,...Rm, and a set of edges. Qk contains an edge (Ri, Rj) iff at the time instant $t(k)$ to which the graph applies some task T holds resource type Ri and requests resource type Rj.

In the system we have R1,...Rm /m=4/ unit capacity resources: the CCWs of each bus control unit, one for each processor. The logic channels correspond to task chains of [2] or to task of [3]. If a logic channel connects processori and processorj i.e. it uses Ri and Rj to execute a transfer then it will be denoted by LCij. An acknowledgement of logic channel LCij means that it holds /say/ Ri and to transfer it requests resource Rj i.e. exactly the edge (Ri, Rj) appears in the allocation graph after an acknowledgement has been sent.

When the transfer takes place edge (Ri,Rj) is deleted and both resources released. As the system contains unit capacity resources a circuit in the allocation graph is a necessary and sufficient condition to deadlock.

Three actions, a request, an acknowledgement and a transfer are needed to transmit a block of user information. The resources required for these actions are allocated by identical resource managers, one for each resource Ri in the system. A resource manager serves the needs of and performs the necessary actions on behalf of the logic channels using its resource. So a resource manager has complete information about the actual resource requirements, and resource usage of the logic channels using the corresponding resource. However, only a part of the allocation state of the complete system is available locally. The decisions made separately by the individual resource managers according to some allocation policy should contribute to the deadlock-free behaviour of the complete

system.

Each resource manager should execute a deadlock prevention algorithm before initiating an action on behalf of a logic channel and the execution should be repeated until nomore action can be initiated. The possibility to initiate an action exists whenever a user command, a request, an acknowledgement is accepted or a transfer is completed. The local information including the resource status is updated automatically by the appropriate procedures. /send request, send acknowledgement, start transfer/. The common bus should be acquired before any decision is taken by a resource manager to execute a possible action. Thereby only a single decision together with the corresponding action is allowed in the complete system at any instant and the consistency of the local information is preserved. Bus ownership ensures exactly the necessary mutual exclusion.

4. Deadlock prevention in MUSCLE

Two algorithms are proposed. The basis of both algorithms is some kind of ordering of the resource usage common with most - if not all - non-preemptive deadlock preventation algorithms. The first algorithm solves the deadlock preventation problem by ordering the actions allowed and thereby indirectly the resource acquisitions.

Two actions involve resource acquisition according to the transfer procedure: acknowledgement and transfer start, both require the CCW to load the transfer parameters. If transfer start precedes acknowledgement deadlock is prevented. This

algorithm works regardless whether the logic channels have priorities or not. It should be noted that the ordering of sending out requests is in fact immaterial because it doesn't mean resource acquisition. Selection of a logic channel among the candidates within any of the three action class can be based on any selection policy: e.g. priority or FIFO. If channels are selected according to their priority permanent blocking may occur. If the selection policy is FIFO, a FCFS /first came first served/ service strategy is realized on user level.

A proof of the deadlock-free behaviour of the system can be based on the following observation. Resource Ri can only be allocated iff no transfer can be started. An edge directed from vertice Ri appears in the allocation graph only iff vertice Ri is isolated, so no circuit may exist.

The second algorithm introduces the notion of priority. The priority of logic channel LCij is defined as follows:

p:if i > j then i.j else j.i end

where . (dot) is the concatenation operator.

```
ALGORITHM1.

begin
  if resource Ri is free
    then select a logic channel where acknowledegement was
           received;
      if successful
        then start transfer
        else select a logic channel where request and user
             command was received
          if selection is successful
            then send acknowledgement
            else select a logic channel where user command
                 was received
                 if successful
                   then send request
                 end
          end
    end
   else select a logic channel where user command was
        received
    if successful
      then send request
    end
 end
end algorithm1;
```

This definition gives a partial ordering among the logic channels, nevertheless it is sufficient because it is based on the full ordering of the resources.

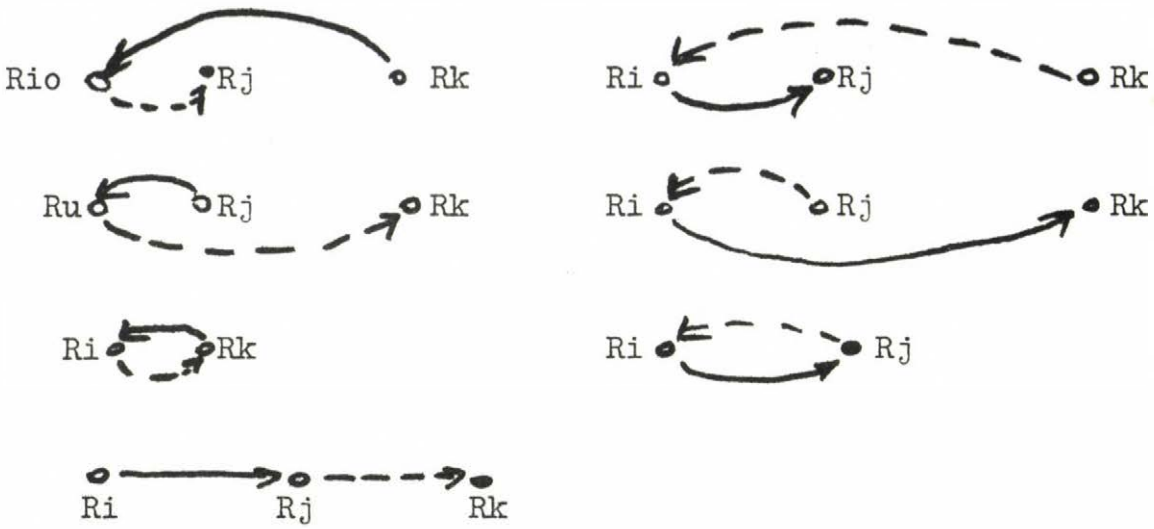
If the indices of the resources are regarded as resource priorities then Ri is called the higher priority and Rj the lower priority resource of logic channel LCij provided that $i > j$. The algorithm prevents deadlock by precluding any resource manager to allocate its resource as the higher priority one to any logic channel if there exists a logic channel which has already acquired or will acquire its lower priority resource. An essentially identical algorithm can be constructed if higher is changed to lower and lower is changed to higher in the last sentence. The precluded case is a necessary condition of the circular wait. Precluded allocation graph configurations are shown on Fig.3. The following algorithm assumes that an artificial logic channel of highest priority is predefined where no.action is necessary at any time.

It is interesting to note that sending out a request is also restricted in certain cases, although it doesn't mean resource allocation.

A correctness proof can be based again on the properties of the allocation graph. An edge directed from vertice Ri to vertice Rj where $i > j$ appears in the graph iff vertice Ri is isolated and will remain isolated until edge (Ri,Rj) is deleted, so no directed path may close. ALGORITHM2 is more involved than the

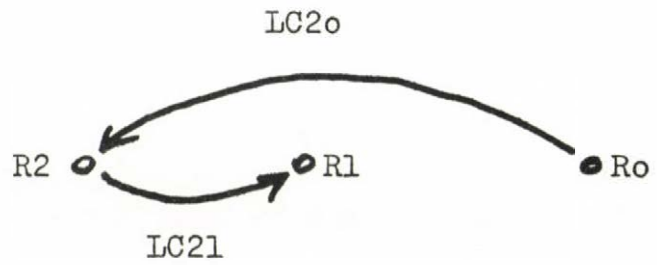
```
ALGORITHM2:
begin
  exit:=false;
  start from the highest priority logic channel
  repeat select next lower priority logic channel
    if resource Ri is free
      then if an acknowledgement was accepted on the
              logic channel selected
           then start transfer;
                exist:=true
           else if a request and a user command was
                   accepted on the logic channel selected
                   then if  resource Ri will be acquired by
                            the selected logic channel as
                            the lower priority one
                            or no resource Rj,  j<i has been
                            or will be acquired by any logic
                            channel served by this resource
                            manager
                            then send acknowledgement
                                 exit:=true
                      end
                 else if a user command was accepted on
                            the logic channel selected
                            then send request;
                                 exit:=true
      end
    else if a user command was accepted on the logic
            channel selected
            then if resource Ri was acquired by a logic
                    channel as the lower priority resource
                    or for resource Rj to be requested j>i
                    then send request;
                         exit:=true
               end
    end
  until exit or nomore action is possible
end algorithm2;
```

previous one but giving preference to certain logic channels could be a desirable property.



Allocation graph configurations precluded by ALGORITHM2.

Fig. 3.



A safe allocation graph in the simple system.

Fig. 4.

5. Hardware support

A number of algorithms similar to those presented can be devised. These algorithms are more restrictive than necessary because only locally available information is used. As an example consider a simple system /see Fig.4/ consisting only of resource Ro, R1, R2 and logic channels LC2o and LC21 using resources R2, Ro and R2, R1 respectively. Assume that Ro is allocated to LC2o and R1 is allocated to LC21. Both algorithms presented would prevent this situation to occur although it is completely safe. The information necessary to construct the allocation state of the total system can be collected by using new types of administrative messages. But this process has to be done before each and every allocation decision and the busy periods of the bus would increase unduly. The cure is worse than the illness itself. Some modification of the hardware seems to be desireable. Assume a 2 bit register in each bus control unit. For BCUi the content of this register is j while CCWi is allocated to logic channel LCij otherwise the content- is i. If the contents of these registers are gated into the data bus whenever the bus is free and loaded into status register /hitherto unmentioned nevertheless existing/ of the BCU acquiring the bus when the bus gets busy then the total system state valid in this instant becomes available.

ALGORITHM3 is based on the fact that a circuit in the allocation graph is a necessary and sufficient condition in this case, so an allocation state is either safe or deadlocked.

```
ALGORITHM3;
begin read in total system state from the status register;
      send requests, start transfers according to some
      arbitrary allocation policy;
      if sending an acknowledgement is intended
         then repeat attempt to allocate resource Ri to a
                     logic channel;
                     check the resulting system state
                     using a deadlock detection algorithm;
                     if the allocation is safe
                        the send acknowledgement
                     end
              until a safe allocation is found or
                    nomore allocation is possible
      end
end algorithm3.
```

Now the deadlock detection algorithms of [2], [3] or [4] can be used and even an exhaustive testing with the complexity of n! is feasible since n=4.

The deadlock detection algorithm can also be realized by hardware. There are 4^4=256 possible configurations of the allocation graph, and for each configuration it is possible to determine whether the state is deadlocked or not. If a 256x1 bit read-only memory is addressed by the 8 bit system state, the output of the ROM will give exactly the required information.

Naturally the complexity of the hardware solution grows very fast if the number of resources is increased.

6. Acknowledgement

The authors would like to thank L.Kozma, J.Horváth and P. Békéssy for their valuable contribution.

REFERENCES

1.Békéssy,P.; Horváth, J.; Kozma, L.; Simonfai, L.:
MUSCLE - Többszámitógépes rendszer-monitor
Rendszerterv. SZÁMKI 1868/77.

2.Coffmann,E.G.jr.; Denning, P.J.:
Operating System Theory
Prentice Hall, 1973.

3.Coffmann,E.G.jr; Elphick, M.J; Shoshani, A.:
System Deadlocks
Computing S rveys, June,1971.

4.Holt,R.C.:
Some Deadlock Properties of Computer Systems
Computing Surveys. Sept. 1972.

SOME REMARKS ON THE ALGEBRA OF AUTOMATA

Jürgen Dassow

1. INTRODUCTION AND DEFINITIONS

We can regard a computer as a network of high complexity which consists of a lot of elementary automata. Such elementary automata are chips realizing certain functions or microprograms, disks and so on. It is obvious that we can construct not only one computer but a big variety of automata using the given elementary automata and the allowed interconnections. Therefore it is natural to ask whether or not we can realize a certain input-output-behaviour by a circuit of the elementary automata. The undecidability of this problem is shown by I.M.Kratko in /5/.

Nowadays often we use an automaton as an acceptor, and therefore we have to ask whether or not a certain set of inputs can be accepted by a circuit of the elementary automata. The author proved the undecidability of this problem (/2/,/3/).

In this connection the sets of automata - called Kleene-sets - which can accept any acceptable set of input words are of interest. The purpose of this paper is the study of the number and the complexity of Kleene-sets.

At first we give the formal definitions of the used mathematical model. We consider the set P_k of all finite initial automata $\mathfrak{A} = (X, Y, Z, \delta, \lambda, z_0)$, where X is a cartesian power of the set $E_k = \{0,1,\ldots,k-1\}$ $(k \geq 2)$ and $Y = E_k$.

We introduce the following operations : identification of inputs, substitution (interconnection between the output of an automaton and an input of an other automaton), feedback (interconnection of the output and an input of an automaton), adding and cancelling of fictive inputs.

We say that $\mathfrak{A} = (X, Y, Z, \delta, \lambda, z_0)$ accepts the set S of words over X if there exists a set $Y' \subset Y$ such that

$$\lambda(z_0, p) \in Y' \text{ if and only if } p \in S .$$

By the theorem of Kleene finite automata can accept exactly the set of regular languages. Therefore we define :
A set $M \subseteq P_k$ is called a Kleene-set if

i) M is closed (with respect to the above operations) and
ii) for any natural number $n \geq 1$ and any regular set S over $E_k^{\,n}$ there exists an $\mathfrak{A} \in M$ such that $\mathfrak{A}$ accepts S.

Because a closed set containing a Kleene-set is a Kleene-set too, we get a partial characterization of the Kleene-sets by the minimal Kleene-sets, where a Kleene-set $M \subseteq P_k$ is called minimal if there exists no Kleene-set N such that $N \subset M$.

2. THE NUMBER OF KLEENE-SETS AND MINIMAL KLEENE-SETS

Let $\mathbb{N}$ and $\mathbb{R}$ denote the set of natural numbers and the set of real numbers respectively. Further let $|A|$ be the cardinality of the set A .
By $A(P_k)$ we denote the number of Kleene-sets in P_k . The following theorem is proved in /2/ and /3/ .

<u>Theorem 1</u> $A(P_2) = 3$

$A(P_k) = |\mathbb{R}|$ for $k \geq 3$.

Concerning the number $B(P_k)$ of minimal Kleene-sets in P_k it is only known that $B(P_2) = 2$ and for $k \geq 3$ $B(P_k) \geq |\mathbb{N}|$.
We want to improve this result.
We put

$$T(M) = \left\{ S : \bigvee_{\mathfrak{A} \in M} \mathfrak{A} \text{ accepts } S \right\}$$

and

$$U = \left\{ (E_k^{\,2})^{*}Q : Q \subseteq E_k^{\,2} \right\} \cup \left\{ E_k^{\,*} P E_k : P \subseteq E_k \right\} \cup \left\{ E_k \right\} .$$

With any regular set S and any set $Q \subseteq E_k$ we associate the regular sets

$R_1(S,Q) = \{(q,p) : p \in S,\ q \in E_k^+,\ l(p) = l(q),\ q = aq',\ a \in Q\}$,

$R(S,Q) = (E_k^{n+1})^* R_1(S,Q)$.

Lemma Let S_1 and S_2 be regular sets, and let $M \subseteq P_k$ be a set such that

i) M is closed,

ii) $U \subseteq T(M)$,

iii) $\{R(S_1,Q), R(S_2,Q)\} \subseteq T(M)$ for any $Q \subseteq E_k$.

Then for any $Q \subseteq E_k$ we have

$R(S_1 \cup S_2,Q) \in T(M)$, $R(S_1 \cdot S_2,Q) \in T(M)$ and $R(S_1^+,Q) \in T(M)$.

Proof : Using induction and projections on subtupels of the tupel of coordinates we get that $U \subseteq T(M)$ implies (E_k^n) $(E_k^n)^* Q \in T(M)$ for all $n \geq 1$ and $Q \subseteq E_k^n$.

In figure 1 we give the construction of an automaton accepting $R(S_1 \cup S_2,Q)$. Each automaton in the figure is given by the accepted set S and the accepting set $Y' \subset E_k$.

Figure 2 shows the construction of an automaton accepting $R(S_1 \cdot S_2,Q)$.

Figure 3 shows the automaton accepting $R(S_1^+,Q)$ if $Y_2 = Y_5$. If $Y_2 \neq Y_5$ we have to iterate the construction of figure 3 in order to get an automaton accepting $R(S_1,Q)$.

Theorem 2 For any closed set $M \subseteq P_k$ the following assertions are equivalent :

i) M is a Kleene-set .

ii) $U \subseteq T(M)$.

Proof : i) $\Rightarrow$ ii) is obvious.

ii) $\Leftarrow$ i) Because we have $R(S',Q') = (E_k^{n+1})^* \{(q,a) : q \in Q\}$ for any $S' = \{a\} \subseteq E_k^n$, $Q' \subseteq E_k$ we get $R(S,Q) \in T(M)$ for any regular set S and any $Q \subseteq E_k$ by our lemma. Figure 4 shows an automaton accepting S .

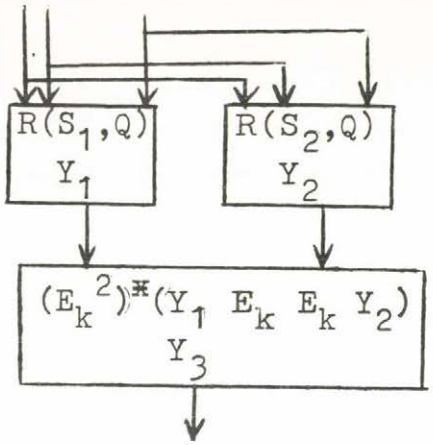


Figure 1

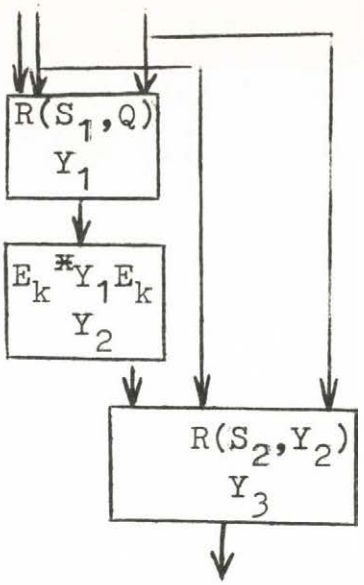


Figure 2

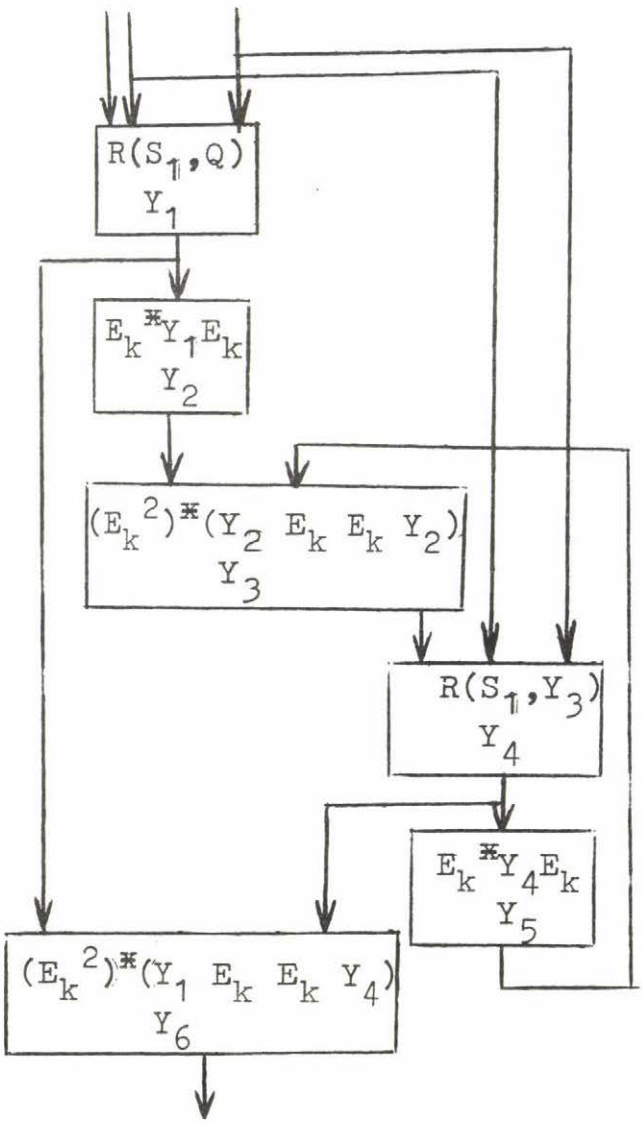


Figure 3

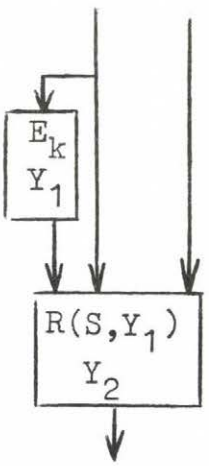


Figure 4

Theorem 3 $B(P_2) = 2$,

$B(P_k) = |\mathbb{N}|$ for $k \geq 3$.

Proof : By the results of /2/ and /3/ we have only to prove that $B(P_k) \leq |\mathbb{N}|$ for $k \geq 3$. The minimal Kleene-sets are different only in the automata $\mathcal{A}(S)$ used to accept $S \in U$. There is only a countable set of automata accepting a given regular set. Thus the finiteness of U implies that the number of minimal Kleene-sets is at most $|\mathbb{N}|$.

Theorem 4 Any set $M \subseteq P_k$ generating a Kleene-set contains a finite set N which generates also a Kleene-set.

Proof : Again let $\mathcal{A}(S)$ denote an automaton accepting S . Using the elements of M we can generate $\mathcal{A}(S)$ for any $S \in U$. Because U is finite we need only a finite set of elements of M for these constructions. By the lemma this finite set generates a Kleene-set.

On the other side it is known that there exist Kleene-sets with a finite basis, Kleene-sets without any basis and Kleene-sets with infinite and without finite basis in P_k , $k \geq 3$ (/4/) .

Theorem 5 Any minimal Kleene-set is finitely generated.

Proof : Let M be a minimal Kleene-set. For any $S \in U$ we have an $\mathcal{A}(S)$ in M . Therefore the closure N of $\{\mathcal{A}(S) : S \in U\}$ is contained in M and is a Kleene-set by the lemma. By the minimality of M we have $N = M$.

3. THE COMPLEXITY OF KLEENE-SETS IN P_2

Concerning generating systems the changing from the whole set P_k to the Kleene-sets was not very successful because we have the undecidability of the problem whether or not P_k or a Kleene-set respectively is generated by a given finite set in both cases. Now we will show that the changing has also

no effect with respect to the complexity.
In P_2 we have only three Kleene-sets, namely P_2, M_{T_0}, M_{T_1}, where M_{T_i} is the set of all automata of P_2 which realize in the first tact a Boolean function f with the property $f(i,i,\ldots,i) = i$. It is easy to see that for any automaton $\mathfrak{A} \in P_2$ we have $\mathfrak{A} \in M_{T_i}$ or $\overline{\mathfrak{A}} \in M_{T_i}$ where $\overline{\mathfrak{A}}$ is defined by

$$\lambda_{\overline{\mathfrak{A}}}(z_0, p) = \text{non}(\lambda_{\mathfrak{A}}(z_0, p)) \quad .$$

Let V be a fixed set generating P_2. Let F be a circuit of elements of V. We define the complexity $l_V(F)$ of the circuit F by the number of elements of V in the circuit. For an automaton $\mathfrak{A}$ we define the complexity $l_V(\mathfrak{A})$ by

$$l_V(\mathfrak{A}) = \min \{ l_V(F) : F \text{ realizes the input-output-behaviour of } \mathfrak{A} \} .$$

It is obvious that there exist a number s such that

$$l_V(\mathfrak{A}) - s \leq l_V(\overline{\mathfrak{A}}) \leq l_V(\mathfrak{A}) + s \quad .$$

With any set $M \subseteq P_2$ we associate the function

$$f_V(n,r,M) = \max \{ l_V(\mathfrak{A}) : \mathfrak{A} = (E_2^n, E_2, Z, \delta, \lambda, z_0), |Z| = r, \mathfrak{A} \in M \}$$

as a measure of complexity of M.
Therefore we get

$$f_V(n,r,M_{T_i}) \leq f_V(n,r,P_2) \leq f_V(n,r,M_{T_i}) + s$$

for any $i \in \{0,1\}$. Thus the complexity functions $f_V(n,r,M)$ are asymptotically equal for all Kleene-sets M.

LITERATURE

/1/ I,M.Copi, C.C.Elgot and J.B.Wright, Realization of events by logical nets. JACM 5, 181-196 (1958) .

/2/ J.Dassow, Kleene-Mengen und Kleene-Vollständigkeit. EIK 10, 287 - 295 (1974) .

/3/ J.Dassow, Einige Bemerkungen zu einem modifizierten Vollständigkeitsbegriff für Automatenabbildungen. Rostocker Mathematisches Kolloquium 3, 69 - 84 (1977) .

/4/ J.Dassow, Ein modifizierter Vollständigkeitsbegriff in einer Algebra von Automatenabbildungen. Dissertation, Rostock 1978 .

/5/ I.M.Kratko, Algorithmic undecidability of the problem of recognizing the completness for finite automata. Dokl.Akad.Nauk 155, 35 - 37 (1964) (in russian) .

/6/ B.A.Trachtenbrot, On the complexity of schemata realized by many-parametric families of operators. Probl. kibernetiki 12, 99 - 112 (1964) (in russian) .

Jürgen Dassow
Department of Mathematics
Wilhelm-Pieck-University
25 Rostock
Universitätsplatz 1
GDR

STATISTICAL INVESTIGATIONS OF CYBER-73 MULTIACCESS SYSTEM

Roman Bednarz*

A special program for performance measurement of SCOPE 3.4 operating system has been described in the paper. The operating system is working on CYBER-73 computer at the Computer Centre CYFRONET - Warsaw. New experiments for channel activity measurements has been prepared, as well as the experiments for the measurements of the coincidence of channel activity and central processor idle state. The results of the disk queue measurements show, that the service time distribution is not a Poisson distribution. Moreover it has been found, that 841 disk does not use the full power of the dual access. As a result of the better distribution of local files between 841 and 844 disk, it was possible to increase the utilization of the central processor by 15 per cent.

INTRODUCTION

Regional Computing Centre "CYFRONET" has been founded in June, 1973. CYFRONET supplies computing power and services to the scientific institutes and universities of Warsaw area. During last four years the initial configuration has been changed many

---

x Institute of Nuclear Research, Regional Computing Centre CYFRONET, Warsaw

times. As a result we came to the mainframe with complicated system of peripheral equipment and heterogeneous terminal network. Because of large demand for the cheap CYFRONET computing power, the center is working seven day a week, three shifts a day. In order to release any reserves of computing power, it was necessary to study the performance of the computing system. Two kinds of performance measurement tools were developed, the first one based on the dayfile analysis and the second based on real time spying of system tables. This paper is devoted to the recent results obtained by second technique.

1. HARDWARE CONFIGURATION

The actual /March, 1977/ configuration of the computer consists of central processor CYBER-73 performing 1.2 million instruction per second /60 bit/, central memory 96 K /K = 1024 words/ and ten peripheral processors with 4 K twelve bit words. The peripheral equipment is attached to 12 channels with the maximum speed 2 millions characters per second. The list of peripheral equipment is the following:

Channel	Equipment
1,2	Dual Access 841 - Multiple Disk Drive, 2 controllers, 7 units 36 mil. character each.
3	844 - Disk, Controller, 2 Units 118 mil. characters each.
5	7077-1 Communications station and 791-1 Communications Controller
6	6671 Multiplexer

10	Display Console
11	Card Punch, Paper Tape Reader /Punch, Plotter with Controllers
12	Card Reader, Printer with Controllers
13	4 nine track 659 Tape Units, 1 seven track 657 Tape Unit with Controller

Five Low Speed Batch Terminal /card reader, printer, console/ and seven TTY are connected to 791-1 Communications Controller through modems. Four PDP-11/45 computers are experimentally connected to Multiplexer through modems. Finally our disks have the following characteristics:

Disk	841	844
Transfer rate in mil. of characters	0.179	0.461
Access Time /miliseconds/:		
Maximum	135	55
Average	75	30
Minimum	25	8
Average Rotational Latency	12.5	8.3

2. DEVELOPMENT OF A SOFTWARE MONITOR

Our computers runs under operating system SCOPE 3.4.1 level 373. Subsystem INTERCOM 4.3 is running all terminals. CIA is a performance measurement tool, which first version was developed by dr. Paul Jalics in 1973 at Stuttgart University. The author of this paper developed in 1974 a new version CIA-CERN, which is independend of computer configuration, suitable for measurements of dual central processor systems and precise measurements of peripheral program utilization. The

version CIA-CERN was described in paper at the National Scientific Conference on MULTI-ACCESS COMPUTER SYSTEMS, Wrocław, June 1975. During 1976 and 1977 a new experiments have been added to CIA-CERN, which resulted in a new version CIA-NRI. The new experiments give the activity of all channels, as well as a coincidence of channel activity and idle state of central processor. Moreover the experiments, measuring the queue length of the disk equipments, has been changed.

CIA-CERN is introduced to the SCOPE 3.4 by a job, which compiles and editlibes three peripheral programs GOG, CIA and GOL. The same job compiles a fortran program CIANIN and catalogues it as a permanent file together with a file CIATXT. Using appropriate operator command we can call CIA to a peripheral processor. CIA can measure permanently or bounce into a peripheral processor every ten seconds. It samples system tables and accumulates results in the form of distribution tables. The tables are contained in the central memory resident program GOG, which is not called to a PP. At the end of measurements the disk resident peripheral GOL is used to write GOG table into a field length of a central program THIRD. THIRD stops the measurements and produces a final report using texts from CIATXT. The report gives the probability distribution of the hardware and software resources utilization, as well as the probability distribution of the computer statuses.

3. CHANNEL ACTIVITY MEASUREMENTS

The experiments $R_0, \ldots R_9$, R_+, R_- measure the probability of channel active state, which is necessary condition for any transfer of data. The typical results for the first production shift are the following /transfers in thousands characters/:

Channel	Equipment	Activity	Transfer Uper Limit	Expected Average Transfer
1	Dual Access 841	0.269	538	48
2	Dual Access 841	0.123	246	22
3	844	0.286	572	131
5	LCC	0.011	22	2
10	Display Console	0.897	1794	
11	CP, PL, TR/TP	0.057	114	
12	PR, CR	0.018	36	2.7
13	4 x NT, MT	0.066	132	4

The Transfer Uper Limit denotes the transfer at maximum channel speed 2 million char/sec. Expected Average Transfer is evaluated as a product of the activity rate and the maximum transfer for the peripheral equipment. During measurements the specification of disk units was following:

Disk 841 — 6 PF units, 1 PUB unit

Disk 844 — 1 SYS unit, 1 PFD unit

where PF - permanent files, PUB - local files

SYS - system files, PFD - permanent files and directory

The measurements refer to standard CDC version of SCOPE 3.4.1 as concerns the distribution of local files between 841 and 844 disks. As we see from table peripheral program DSD, runing the Display

Console, keeps almost all the time channel in active state. DSD permanently reads an information from system tables and buffers. The information is necessary for the renewal of two system displays. High activity can be observed on 844 disks, serving as the system devices. The 844 disks perform very frequent loadings of peripheral programs and compilers overlays, as well as the swaping of central programs. Relatively low activity can be observed on the channel five which links with the communication processor. However we should notice that communication processor LCC works with seven 110 bode TTY`s, and five 2400 bode LSBT, which all together can not produce a substantial transfer. Another set of experiments Q_0, Q_1,..., Q_9, Q_+, Q_- gives the conincidence of the central processor idle state and the channel active state. The measured value of the coincidence Q is compared with the product R.J of the channel active state probability and the idle state probability. The product should be close to Q for the statistically independent events. The results of measurements, concurrent with the previous experiment are listed below:

Channel	Q	R.J
1	0.077	0.059
2	0.034	0.027
3	0.078	0.062
5	0.0025	0.0023
10	0.217	0.195
11	0.014	0.012
12	0.0034	0.0039
13	0.019	0.014

The idle state probability, measured by another experiment, was J = 0.218. Comparing Q and R.J we can see that activity of 841 and 844 disks coincides with idle state about twenty per cent more often than it should coincide statistically independent events. This indicates disk activity as one of the sources of idle time. Peripheral processors activity is likely to be another source of idle time.

4. MEASUREMENTS OF DISK QUEUES

Another set experiments K_0, K_1... K_9 has been changed. Original version of the experiments was written by Dr. Paul Jalics. This version was measuring the number of stack requests, which are linked in chain by the central program SPM. The modified version takes also into account stack requests, which are delinked from the chain by peripheral program 1SP. The 1SP program can delink up to eight stack request in order to choose the best candidate for the disk input/output operations. The requests are stored in the internal 1SP table "pool" and each of them disappears from central memory stack, after input/output operations has been finished. Jalics experiments gave about two times smaller utilization of the disks than it is in fact. The typical measurements, by the modified experiments, are the following:

Experiment K2 for 844 disks:

Number of requests in stack - 1	Probability - P_1
0	0.603
1	0.263
2	0.098

3	0.029
4	0.006
5	0.001

From the above data we can calculate the utilization factor:

$$\rho = 1 - P_0 = 0.397$$

Similarly the average queue length is:

$$L = \sum_{i=2} P_i \left(i - 1 \right) = 0.179$$

Under assumption of the Poisson interarrival distribution we can find the queue length for the equal service time case:

$$L_D = \frac{\rho^2}{2 \left(1 - \rho \right)} = 0.131$$

We see that the measured queue length is larger than it would be for the equal service time case.

Assuming Erlang service time distribution with mean μ^{-1} a free parameter η:

$$dB(t) = \frac{(\mu \eta)^{\eta}}{\Gamma(\eta)} \cdot e^{-\mu t} \cdot t^{\eta - 1} \, dt$$

we can look for agreement with the measurements.

The case $\eta = 1$ corresponds to Poisson service time distribution, which is usually assumed in the theoretical considerations. However in order to get agreement with the measured queue length, we should take $\eta = 2.76$, since the Erlang queue length is

$$L_E = L_D \cdot \left(1 + \frac{1}{\eta} \right)$$

We can say that the real service time distribution is something in between Dirac equal time distribution and Poisson distribution of service times.

The interpretation of the measurements for dual access 841 Disk is a more complicated task.

The Experiment K3 gives the following probability distribution:

Number of request - i	Probability - P_i	Theory
0	0.293	0.251
1	0.218	0.300
2	0.164	0.179
3	0.126	0.107
4	0.094	0.064
5	0.061	0.038
6	0.031	0.023
7	0.010	0.013
8	0.002	0.008

We can consider dual access 841 Disk as multiserver consisting of two servers $M = 2$. In this case the utilization factor is given by the formula:

$$\rho = M \cdot (1 - P_0 - P_1) + P_1 = 1.196$$

So far the only one exact solution of multiserver is known for the Poisson arrival and service distributions.

The solution has form

$$P_i = \frac{1}{i!} \rho^i \quad /P \qquad i = 0,1,\ldots,M$$

$$P_i = P_M \left(\frac{\rho}{M}\right)^{i-M} \qquad i = M+1,\ M+2,\ldots$$

$$P^{-1} = \sum_{i=0}^{M-1} \frac{1}{i!} \rho^i + \frac{1}{M!} \frac{\rho^M}{1 - \frac{\rho}{M}}$$

The theoretical results for M=2, ρ = 1.196 are listed in the last column of the above table.

The queue length is given by the formula:

$$L = \sum_{i=3}^{\infty} P_i (i - 2)$$

The above formula gives the measured queue length $L_M = 0.681$ and the theoretical queue length $L_P = 0.554$. In fact this is a surprise, since for 841 Disk with wingle access /p = 0.733/ the measured queue length $L_M = 1.145$ is much smaller than the theoretical queue length Poisson arrival and service distributions $L_P = 2.012$.

Let us notice that the ratio of second access activity to first access activity

$$\frac{R_2}{R_1} = 0.457$$

is much smaller in comparison with the value for multiserver with M = 2. This value is the ratio of the probability of two and more requests to the probability of one and more requests:

$$\frac{1 - P_0 - P_1}{1 - P_0} = 0.681$$

It means that second access works only 66 per cent of time, which we would expect.

First reason of reduced second access activity is relatively simple. Second access is not working, when we have more than one

request for one unit only. Such a situation happens relatively frequent for the unit specified as PUB. We can reduce this effect by spreading the local files over all disk units. Another reason is more complicated. 1SP delinks the requests from SPM chain in order to choose the best request. 1SP can call another 1SP called PARTNER if it has something to do for second access. However it may happen, that there is nothing to do for PARTNER according 1SP internal LOOP table, but during 1SP administrative and input/output activities SPM has built a new chain of requests. The new chain my contain the requests appropriate for PARTNER, but PARTNER must wait until main 1SP will finished with input/output. Such multiserver is in fact party blind, because PARTNER as a slave can not see independently the request chain in central memory. In order to overcome the blindess of multiserver both 1SP should read the request chain independently, which means that both 1SP should permanently resident in peripheral processors.

The measurements of disk queues has shown very unfortunately high utilization of 841 disks, which have approximately 2.5 times lower transfer and 2.5 times higher access time than 844 disk. The utilization factor per server was 0.598 for 841 disk and only 0.397 for 844 disk. Similarly the queue length was 0.681 for 841 disk and only 0.178 for 844 disk.

The investigations, performed by the author and Mrs. Danuta Mąkosa, were concentrated on the mechanism of file opening. In order to find a record block for new or overflowing local file, peripheral program 3DO searches the disk equipment with the lowest Activity. From the equipment, 3DO takes the unit with the heighest number of free record blocks. The Activity is stored for each disk equipment in a byte of Device Activity Table and it is calculated every second by 1RN form the formula:

$$ACTIVITY = \frac{1}{2} \cdot \left(OLDACTIVITY + NEWACTIVITY \right)$$

$$NEWACTIVITY = \left[S + \left(S + SPEED \right) \cdot COUNT \right] \; / \; D$$

where

for 841 disk

S = 0 /No System/, D = 2 /dual access/

and for 844 disk

S = 1 /System/, D = 1 /single access/

COUNT is a number of disk requests during last second.

For standard SCOPE 3.4.1 system 1RN contains:

for 841 disk

SPEED = 8

for 844 disk

SPEED = 4

Since 844 disk contains the operating system, the value of COUNT is usually high. Therefore local files were opened almost entirely on 841 public unit. In order to increase ACTIVITY of 841 disk we have changed 1RN taking:

for 841 disk

SPEED = 20

for 844 disk

SPEED = 2

As a result of 1RN changes, the utilization factor and the queue length of 844 disk increased from 0.397 to 0.476 and from 0.178 to 0.345 respectively. The utilization and queue length of 841 disk decreased dramatically from 1.196 to 0.660 and from 0.685 to 0.085. Substantial increase of central processor utilization by users was observed from 0.647 to 0.795, which roughly corresponds to 23 per cent increase of throughput.

6. CONCLUSIONS

The advanced software monitors proved to very useful from both theoretical and practical view point. First of all it is clear that we can not assume Poisson arrival and service distributions for disk queues under SCOPE 3.4 operating system. This eliminates computer models, which are the easiest for analitical solution. It is very likely that some aspect of CYBER computers can be described by the Diffusion Approximation to Queining Networks, which is built on less restrictive statistical assumptions. The author is involved in the development of the Diffusion Approximation for Multiserver Networks and he try to present the results on future conferences. In order to built a consistent CYBER model much more statistical information should be collected, that it is possible by means of CIA. We need for example transitions probabilities between different servers for a job. We should also know about paralell use of servers by a job. Questions may lead to model oriented software monitors, which should supply constants for models. A part of importance for computer models, which are useful mostly for installation planning, the measurements by software monitors give us the utilization map of computing system. Such an information can easly suggest us the necessary changes in hardware configuration or operating system. The performance measurements are especially valuable for the complicated computing systems, where quantitative analysis may lead to serious throughput increase.

REFERENCES

1 B e d n a r z R., A New Version of CIA, Proceedings of the ECODU-XVIII, London, September 1974.

2 B e d n a r z R., Problems of the software maintenance and performance measurements of CYBER-72 computer at CYFRONET - Warsaw. The National Scientific Conference on MULTI-ACCESS COMPUTER SYSTEMS, Wrocław, 1975, p. 49-58.

3 B e d n a r z R., On the optimization of CYBER-72-16 multiaccess computer. Conference on the MULTIACCESS PROBLEMS IN COMPUTER SYSTEMS, Międzygórze, 1976, p. 11-21.

ON A QUEUEING PROBLEM IN THE THEORY OF OPERATING SYSTEMS

L. Lakatos

In [1] there are considered systems in which the jobs are given preferential treatment based on priorities associated with jobs. There is assumed that the priority of a job is an integer fixed at arrival time. For the service discipline it is assumed that whenever a job completed the processor is next assigned to that job at the head of the highest priority /lowest level/ nonempty queue. Once a job is begun on a processor it is allowed to run to completion. Independent Poisson arrivals are assumed for the different priority classes with arrival rate λ_i $/i=1,2,\dots,n$ /. Arbitrary possibly different service time distributions exist for the priority classes, the corresponding distribution functions will be denoted by $B_i(x)$ for the i-th class. In [1] the mean waiting time of random arrivals to the i-th queue and the mean waiting time in queue for the jobs of priority i during statistical equilibrium are computed.

Here we consider a service system where the number of priority classes n and the number of present jobs k are restricted. We want to have the stationary probabilities to service a job of a given type or to be in free state and to get the busy period's distribution law.

Under such restrictions the functioning of our system may be described by means of a certain piecewise-linear process, introduced in [3]. By Kovalenko's theorem if the number of states of the piecewise-linear process is finite and the expectations of each continuous component are finite, too, then the sought-for stationary distribution exists. Now we will follow the way, described in [2]. In such case accord-

ing to §4.11 of [3] the sought-for probabilities may be got by the formulas:

$$p_0=\frac{t_0}{t_0+t_1+\ldots+t_n}\,,\quad p_1=\frac{t_1}{t_0+t_1+\ldots+t_n}\,,\ \ldots\,,\ p_n=\frac{t_n}{t_0+t_1+\ldots+t_n}\,,$$

where t_0 is the expectation of free state and $t_i\ /\ i=1,2,\ldots,n$ / the expectation of service time for i -th type's jobs for a busy period. It is clear

$$t_0=\frac{1}{\lambda_1+\lambda_2+\ldots+\lambda_n}\,.$$

To get $t_i\ /\ i=1,2,\ldots,n$ / we determine the busy period's distribution law. Let us denote it by $K(x)$, then it may be written in the form

$$K(x)=\sum_{i=1}^{n}\frac{\lambda_i}{\sum_{j=1}^{n}\lambda_j}K_i(x)\,,$$

where $K_i(x)$ is the desired distribution law in case when the service starts with an i -th type's job. Let $K_{i_1\ldots i_j}(x)$ $/\ i_1,\ldots,i_j=1,\ldots,n;\ j=1,2,\ldots,k$ / denote the distribution function of the busy period if at the moment when service starts in the system there are present j jobs and they have entered the system in such order. At first we deal with the case when the service runs according to the FIFO rule, i.e. the jobs are serviced in the order of their occurence. In this case the functions may be got from the next system of equations:

$$K_i(x)=\int_0^x e^{-\lambda y}\,dB_i(y)+$$

$$+\sum{}'\sum_{l=2}^{k-1}\int_0^x K_{i_2\ldots i_l}(x-y)P_{i_2\ldots i_l}(y)\,dB_i(y)+$$

$$+\sum{}''\int_0^x K_{i_2\cdots i_k}(x-y)P_{i_2\cdots i_k}(y)\,dB_i(y),$$

$$i=1,2,\cdots,n\ ,$$

where

$$P_{i_2\cdots i_l}(y)=e^{-\Lambda y}\frac{y^{l-1}}{(l-1)!}\prod_{j=2}^{l}\lambda_{i_j}\quad\text{and}$$

$$P_{i_2\cdots i_k}(y)=\frac{1}{\Lambda}\prod_{j=2}^{k}\lambda_{i_j}\left\{\int_0^y e^{-\Lambda u_{k-1}}\frac{u_{k-1}^{k-3}}{(k-3)!}\,du_{k-1}-\right.$$

$$\left.-e^{-\Lambda y}\frac{y^{k-2}}{(k-2)!}\right\};$$

$$K_{i_1\cdots i_m}(x)=\int_0^x K_{i_2\cdots i_m}(x-y)e^{-\Lambda y}\,dB_{i_1}(y)+$$

$$+\sum{}'\sum_{l=m+1}^{k-1}\int_0^x K_{i_2\cdots i_m i_{m+1}\cdots i_l}(x-y)P_{i_{m+1}\cdots i_l}(y)\,dB_{i_1}(y)+$$

$$+\sum{}''\int_0^x K_{i_2\cdots i_m i_{m+1}\cdots i_k}(x-y)P_{i_{m+1}\cdots i_k}(y)\,dB_{i_1}(y),$$

where

$$P_{i_{m+1}\cdots i_l}(y)=e^{-\Lambda y}\frac{y^{l-m}}{(l-m)!}\prod_{j=m+1}^{l}\lambda_{i_j},$$

$$P_{i_{m+1}\dots i_k}(y)=\frac{1}{\Lambda}\prod_{j=m+1}^{k}\lambda_{i_j}\left\{\int_0^y e^{-\Lambda u_{k-1}}\frac{u_{k-1}^{k-m-2}}{(k-m-2)!}du_{k-1}-\right.$$

$$\left.-e^{-\Lambda y}\frac{y^{k-m-1}}{(k-m-1)!}\right\},$$

$$1\leqslant i_1,i_2,\dots,i_m,i_{m+1},\dots,i_k\leqslant n\ ,\quad 2\leqslant m\leqslant k-2\ ;$$

$$K_{i_1i_2\dots i_{k-1}}(x)=\int_0^x K_{i_2\dots i_{k-1}}(x-y)e^{-\Lambda y}\,dB_{i_1}(y)+$$

$$+\sum_{j=1}^{n}\frac{\lambda_j}{\Lambda}\int_0^x K_{i_2\dots i_{k-1}j}(x-y)\left[1-e^{-\Lambda y}\right]dB_{i_1}(y)\ ,$$

$$1\leqslant j\leqslant n.$$

Here $$\Lambda=\lambda_1+\lambda_2+\dots+\lambda_n\ ,$$

and $\sum'$ and $\sum''$ mean that the summation is extended over all possible different values of indices. $P_{i_1\dots i_m}(y)$ gives the probability that during y m new jobs enter our system in sequence determined by $i_1, i_2,\dots, i_m$.

It is easily seen that we have a system of Volterra integral equations, which under certain restrictions may be

solved using the Laplace-Stieltjes transform. For us it is enough to get expressions only for $K_i(x)$, as at the beginning of service in our system there is only one job. Computing the expectation of the busy period on the basis of its Laplace-Stieltjes transform we can determine the time durations, which we try for the service of jobs of different types.

The number of equations in the above system is

$$\sum_{j=1}^{k-1} n^j$$

If we assume that in our system the service runs according to the priorities of jobs, then at the starting moments of different jobs' service they are already arranged by their priorities independently of the order of their occurence in the system. So the cases which are different only in the order of occurence are considered as identical and the corresponding probabilities must be summed up. Of course the number of equations will be less, it is given by the next easily verifiable by induction recurrence relation

$$\sum_{j=1}^{k-1} S_n^{(j)} \quad , \text{ where}$$

$$S_i^{(m)} = S_1^{(m-1)} + S_2^{(m-1)} + \ldots + S_i^{(m-1)} \quad ,$$

$$S_i^{(1)} = i \, .$$

References

[1] Coffmann E.G.,Denning P.J.:Operating Systems Theory, Englewood Cliffs, Prentice Hall, 1973.

[2] Lakatos L.:Vizsgálatok az M/G/1 tipusú tömegkiszolgálási rendszerek körében, egyetemi doktori értekezés, ELTE TTK, 1977.

[3] Гнеденко Б.В.,Коваленко И.Н.:Введение в теорию массового обслуживания, изд.Наука, М., 1966.

MULTIFREQUENCY ALOHA-TYPE SYSTEMS

by

Otto SPANIOL

Institut für Informatik
der Universität Bonn
Wegelerstrasse 6
D - 5300 Bonn
(W.-Germany)

Summary: In this paper we present a modification of the (slotted) ALOHA network model. It will be shown that the maximum achievable throughput is significantly higher than e^{-1} (the theoretical bound derived by Abramson) by providing several user channels together with a queue of maximum length L and several transponder frequencies for packet retransmissions.

Furthermore the stability problem for multifrequency networks will be discussed. It turns out that the communication system is stable if the retransmission control policy derived by Fayolle, Gelenbe and Labetoulle [FGL] is applied (i.e. the retransmission probability of a blocked terminal should be inversely proportional to the number of blocked stations).

The last section of the paper deals with the analysis of the 'dominating channel model' which may be regarded as a special case of multifrequency networks.

I. INTRODUCTION

Packet radio transmission is an interesting alternative to wire communication networks (see [KL] for a summary of some advantages of such networks). In the last few years the properties of these models have been analyzed by several authors.

In this paper we restrict ourselves on the particularly interesting slotted ALOHA type systems which combine many terminals by means of a user frequency for the transmission of user packets and a second frequency which will be used either for retransmissions from a transponder (central station or satellite) to the

destination terminal or for the acknowledgment traffic. All packets are of constant length and are transmitted over an assumed noiseless channel, i.e. errors caused by random noise are neglected compared to packet interferences [T]. If after a finite time-out no ack is obtained (in satellite systems the users can listen their own transmissions, thus no special acknowledgments are necessary and the time-out is given by the round trip delay) then the transmission has been colliding with other transmissions. In this case it will be assumed that none of the interfering packets will be correctly received (for a modification of this assumption see section V) and the corresponding terminals are blocked: after a random retransmission delay the packet transmission has to be repeated until it achieves success. After a successful transmission a terminal will be active again, i.e. it begins with the generation of a new packet (cf. figure 1).

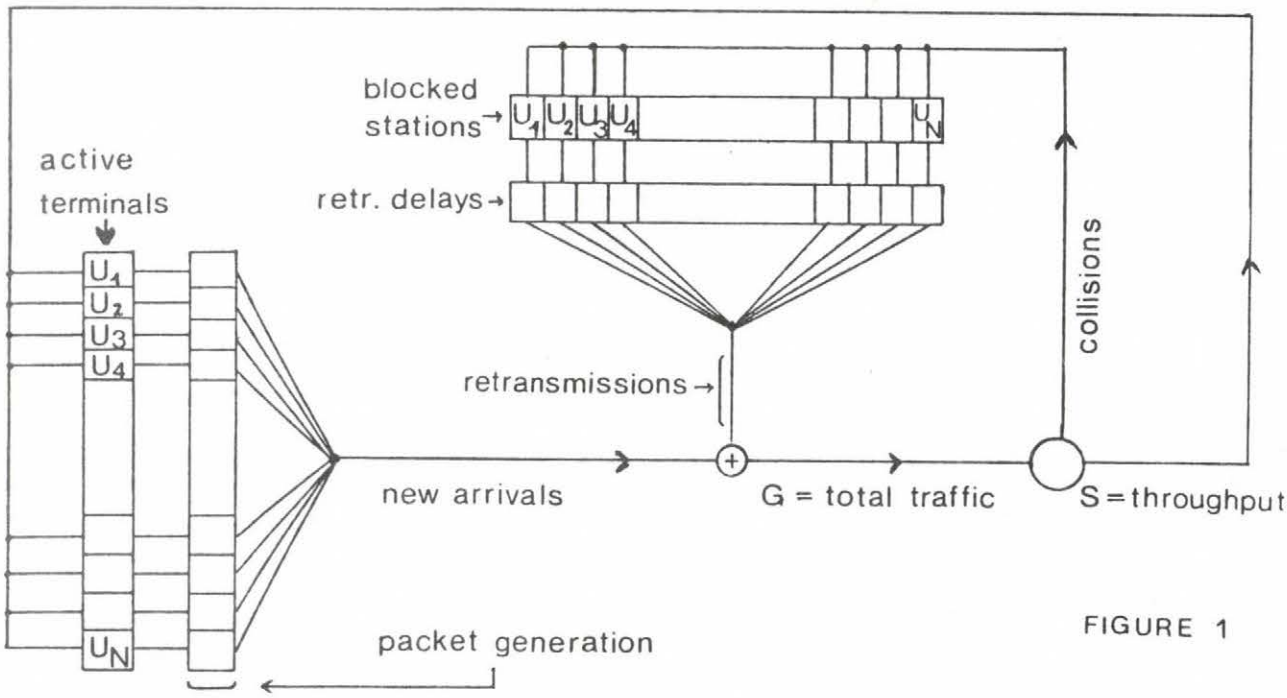


FIGURE 1

II. THROUGHPUT AND STABILITY PROBLEMS OF ALOHA SYSTEMS

Under simplifying assumptions there is the following simple tradeoff between the throughput S and the channel traffic G of a slotted ALOHA system:

Theorem 1: If the channel traffic and the throughput of a slotted ALOHA system are Poisson processes with rates G and S respectively, then

$$S = G \cdot p_0 = p_1 = G \cdot e^{-G}$$

where p_i = Pr(i transmissions in a slot) = $\frac{G^i}{i!} \cdot e^{-G}$.

It has been shown by Fayolle et al. [FGL] that the Poisson assumptions may only be satisfied for short time intervals since the slotted ALOHA system is unstable, i.e. there is no stationary positive throughput for uncontrolled ALOHA networks.

Due to variations in channel traffic intensity the number of blocked stations will be increased; this will result in higher retransmission rates (if neither the newly arriving packets nor the retransmissions are controlled). In connexion with such a situation it will be almost certain that most of the retransmissions will result in new collisions; the number of blocked terminals will be further increased and the system 'breaks down'.

On the other hand, it has been shown in [FGL] that the ALOHA channel may be stabilized if the retransmission probability is controlled by the number of blocked terminals:

Theorem 2 [FGL]:

Let $g_i(n) := \binom{n}{i} \cdot f(n)^i \cdot (1-f(n))^{n-i}$

$= \Pr(\text{i blocked terminals retransmit} \mid \text{n terminals are blocked})$

$c_i := \Pr(\text{i active terminals transmit in a slot})$

where $f(n)$ is the retransmission probability of a blocked station.
Troughout this paper the infinite user model will be supposed, i.e. the rate of new arrivals is independent of the number of blocked stations.

Then:

a. $S_n(f) = \Pr(\text{successful transmission} \mid \text{n blocked})$

$= \Pr(\text{exactly one transmission per slot} \mid \text{n blocked})$

$= c_o \cdot g_1(n) + c_1 \cdot g_o(n)$

b. $\max_f S_n(f) = c_o \cdot \left(\frac{n-1}{n-a}\right)^{n-1}$ is obtained for

$f(n) = f^*(n) = \frac{c_o - c_1}{nc_o - c_1} = \frac{1-a}{n-a}$ where $a := \frac{c_1}{c_o}$

i.e. $\max_f S_n(f) = c_o \cdot \left(1 + \frac{a-1}{n-a}\right)^{n-1} \xrightarrow[n\to\infty]{} c_o \cdot e^{a-1}$

c. If the new arrivals are Poisson (i.e. $c_i = \lambda^i/i! \cdot e^{-\lambda}$, $a = \lambda$) then

$\max_f S_n(f) \longrightarrow e^{-\lambda} \cdot e^{\lambda-1} = e^{-1}$

and e^{-1} is a <u>lower</u> bound for the conditional throughput $S_n(f)$.

Thus the throughput of an ALOHA system which is controlled by the retransmission control policy $f^*(n)$ is given by:

$$S(f) = \sum_{i=o}^{\infty} \Pr(\text{i terminals are blocked}) \cdot S_i(f) \geq e^{-1} .$$

See [FGL] for a proof of theorem 2. The proof can also be obtained as a special case of theorem 4.

The optimal retransmission probability is inversely proportional to the number of blocked terminals; this means intuitively that the expected channel traffic produced by blocked terminals is bounded by $1-\lambda$ where $\lambda = c_1/c_o$ is the expectation of new arrivals:

$$n \cdot f^*(n) = n \cdot \left(\frac{1-c_1/c_o}{n} + O\left(\frac{1}{n^2}\right) \right) = 1 - c_1/c_o + O\left(\frac{1}{n}\right) = 1 - \lambda + O\left(\frac{1}{n}\right) .$$

III. THE (m,d,L)-ALOHA SYSTEM

In this section it will be shown that the throughput of a slotted ALOHA system is significantly increased if several user frequencies are used.

The study of these modified ALOHA systems has been initiated by the obvious conjecture that the number of user packet collisions will be reduced by utilizing two or more user frequencies. If more packets arrive than the transponder (satellite) is able to retransmit in a given slot, then these additional packets have to wait in a FIFO-queue of maximum length L ($o \leq L < \infty$)which will be filled if the system is temporarily overcommitted and will be emptied during periods of lower load; thus, this mechanism could be able to smooth a substantial part of systems load variations. The finite length of the queue corresponds to a finite time-out after which a sender assumes that his packet has had a conflict with one or more other packets. If more packets arrive than the queue can accept then some of them are lost. Furthermore it will be clear that the mean queue length will be shortened if additional transponder frequencies are foreseen, but certainly it is of no use to utilize more transponder frequencies than user channels.

The outlined model is called (m,d,L)-ALOHA system,

where m ($m \geq 1$) number of user frequencies
d ($1 \leq d \leq m$) " " retransmission frequencies
L ($L \geq o$) maximum length of the queue.

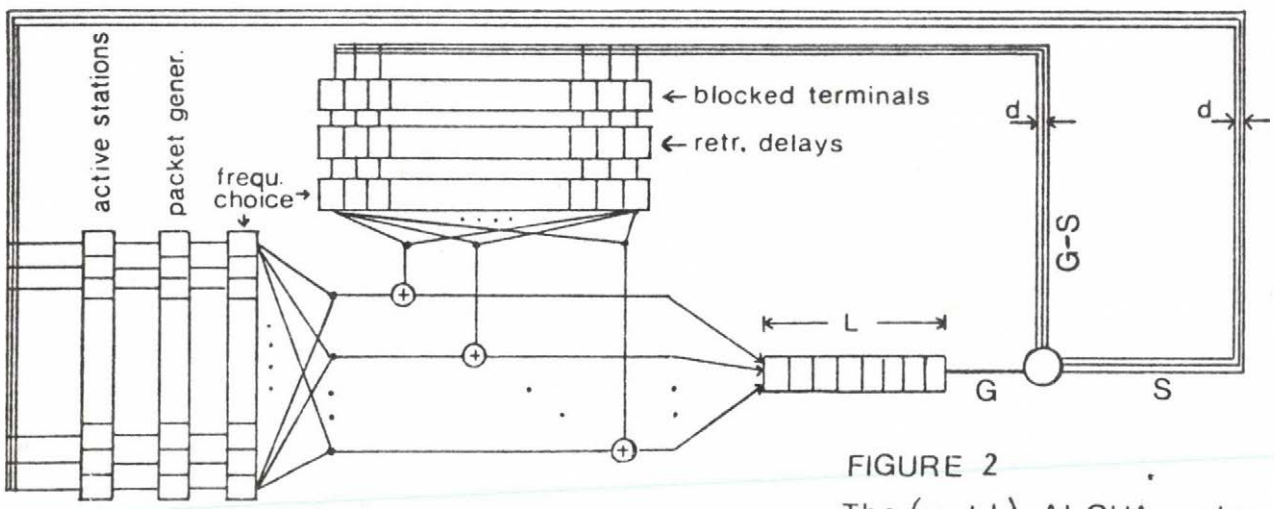


FIGURE 2
The (m,d,L)-ALOHA system

The queue will be organized as follows (L^+ := actual queue length):

Number of occupied channels	Change of queue size	Comments
$o \leq k \leq d - L^+$	$L^+ := o$	All of the waiting packets and all new packets are retransmitted
$d-L^+ < k \leq d+L-L^+$	$L^+ := L^+ + k - d$	All packets are accepted; the oldest packets (as well as $d-L^+$ new packets, if $L^+ < d$) are retransmitted; the rest of the new packets is accepted in the queue
$d+L-L^+ < k \leq m$	$L^+ := L$	$k - (d+L-L^+)$ packets are lost; the probability that a packet is not accepted is given by $p = (k - (d+L-L^+))/k$

Remarks:

a. The 'intelligency' of the transponder consists in recognizing channels which do not contain any (collided or correct) information.

b. Colliding packets are not distinguished from channels which are occupied by a single (i.e. undisturbed) packet.

c. Instead of choosing equal probabilities, it is possible to introduce packet loss probabilities favouring some of the user frequencies, but this will not result in a higher throughput if all channels have equal transmission rates, i.e. if Pr(terminal i transmits over channel k) = 1/m for all i and all k.

Theorem 3 is a generalization of theorem 1; it depends on the simplifying assumption that a stationary positive throughput exists. The stability problem (derivation of a retransmission control policy which is able to stabilize the system) will be studied in section IV of this paper.

Theorem 3: Consider a (m,d,L)-ALOHA system where the user channels are independently used for transmissions (each of them with probability 1/m).
Assume that the throughput and the total traffic af the system are Poisson with parameters $S = S(m,d,L)$ and $G = G(m,d,L)$ respectively.

Then:

$$S = G \cdot e^{-G} \cdot \sum_{k=1}^{m} \binom{m-1}{k-1} \cdot (e^{G/m}-1)^{k-1} \cdot a_k$$

where a_k := Pr(the information arriving on channel j is accepted | k of the m channels are occupied)

$$= \Pr(k \leq d+L-L^+) + \sum_{h=o}^{k-d-1} \Pr(L-L^+=h) \cdot \frac{d+h}{k} = 1 - \sum \Pr(L-L^+=h) \cdot (k-d-h)/k \ .$$

Proof: Since the total traffic is a Poisson process, it is composed of m independent Poisson processes of rate G/m each.

Thus: Pr(channel j is idle) $= e^{-G/m}$

Pr(channel j is occupied) $= 1 - e^{-G/m}$

Pr(exactly one packet on channel j) $= G/m \cdot e^{-G/m}$.

Since all channels are equally loaded we come out with:

$$S = \sum_{j=1}^{m} \sum_{k=1}^{m} \Pr(\text{exactly one packet arrives on channel } j \wedge k-1 \text{ out of the other } m-1 \text{ frequencies are occupied} \wedge \text{the packet arriving on channel } j \text{ is accepted})$$

$$= m \cdot \sum_{k=1}^{m} G/m \cdot e^{-G/m} \cdot \binom{m-1}{k-1} \cdot (e^{G/m}-1)^{k-1} \cdot a_k$$

$$= G \cdot e^{-G} \cdot \sum_{k=1}^{m} \binom{m-1}{k-1} \cdot (e^{G/m}-1)^{k-1} \cdot a_k$$

The following applications of theorem 3 show that the throughput is increased by extending the number of frequencies and/or the maximum queue length.

Examples:

1. m = d = 1 :

$$S(1,1,L) = G \cdot e^{-G} \qquad \text{(cf. theorem 1).}$$

2. m = 2 , d = 1 (i.e. $a_1 = 1$, $a_2 = 1 - \frac{1}{2} \cdot \Pr(L^+=L)$) :

$$S(2,1,L) = G \cdot e^{-G} \cdot (1 + (e^{G/2}-1) \cdot (1 - \tfrac{1}{2} \cdot \Pr(L^+=L)))$$

$$= G \cdot (e^{-G/2} - \tfrac{1}{2} \cdot \Pr(L^+=L) \cdot (e^{-G/2} - e^{-G}))$$

3. m = d ($a_i = 1$ for i = 1,...,m) :

$$S(m,m,L) = G \cdot e^{-G} \cdot \sum_{k=1}^{m} \binom{m-1}{k-1} \cdot (e^{G/m}-1)^{k-1}$$

$$= G \cdot e^{-G/m} .$$

This system can also be understood as a parallel combination of m independent (1,1,L)-systems with rate G/m each, thus

$S(m,m,L) = m \cdot (G/m \cdot e^{-G/m}) = G \cdot e^{-G/m}$, confirming the result.

4. d = 1 , L = o (i.e. $a_i = \frac{1}{i}$ for i = 1,...,m) :

$$S(m,1,o) = G \cdot e^{-G} \cdot \sum_{k=1}^{m} \frac{1}{k} \cdot \binom{m-1}{k-1} \cdot (e^{G/m}-1)^{k-1}$$

$$= G/m \cdot (1 - e^{-G}) / (e^{G/m} - 1).$$

A plot of this relationship for several values of m is given in figure 3.

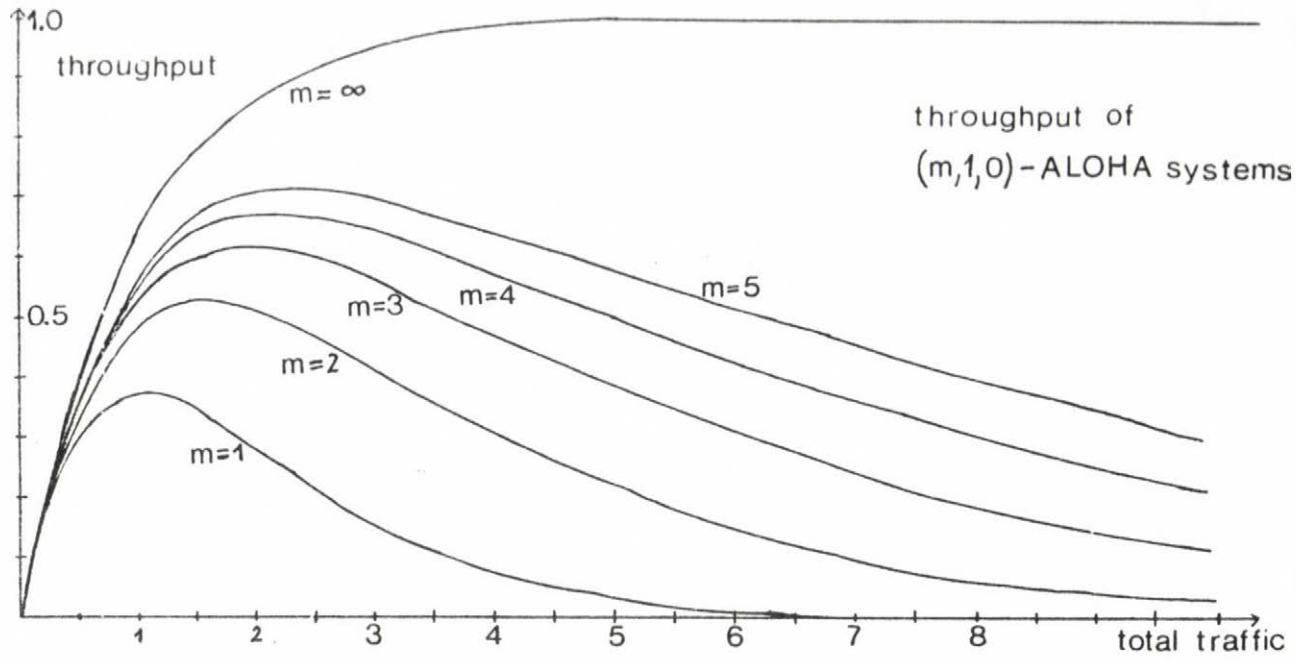


FIGURE 3

5. $d = m-1$ (i.e. $a_i = \begin{cases} 1 & \text{for } i < m \\ 1 - 1/m \cdot \Pr(L^+=L) & \text{for } i = m \end{cases}$) :

$$S(m,m-1,L) = G \cdot \left(e^{-G/m} - \frac{1}{m} \cdot \Pr(L^+=L) \cdot (e^{G/m}-1)^{m-1} \right)$$

$$S(m,m-1,o) = G \cdot e^{-G/m} \cdot \left(1 - \frac{1}{m} \cdot (1-e^{-G/m})^{m-1} \right) .$$

For a given total traffic G the following general relations are obtained ([Sp1]):

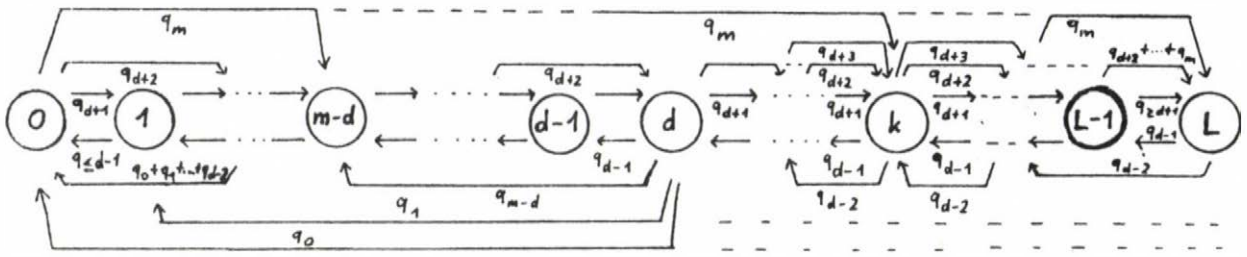
$S(m,d,o) \leq S(m+1,d,o)$ with equality iff $G = o$

$S(m,d,L) \leq S(m,d,L+1)$ " " " $G = o \wedge m = d$

$S(m,d,L) \leq S(m,d+1,L)$ " " " $G = o \wedge m = d$.

Open question: $S(m,d,L) \overset{?}{\leq} S(m+1,d,L)$ for $o < L < \infty$.

The parameters a_k of theorem 3 contain the probabilities $\Pr(L^+=i)$; these values are obtained from a state diagram by means of the flow conservation method (see figure 4). The details can be found in [Sp1].



where q_i := Pr(i out of the m channels are occupied)

$$= \binom{m}{i} \cdot (1 - e^{-G/m})^i \cdot (e^{-G/m})^{m-i}$$

FIGURE 4

IV. STABILITY PROBLEMS FOR (m,d,L)-ALOHA SYSTEMS

The Poisson assumptions of theorem 3 may not be satisfied in reality for the same reasons as in section II.

As a generalization of theorem 2 it will be shown in this section that (m,d,L)-systems may be stabilized by a retransmission control policy which turns out to be somewhat similar to the corresponding strategy proposed by Fayolle et al. ([FGL]).

For a (m,d,L)-ALOHA system we define the following events (no channel will be favoured, thus the channel index j may be suppressed):

$W_{i,j}(n) = W_i(n) \longleftrightarrow$ i of the n blocked terminals retransmit on channel j

$T_{i,j} = T_i \longleftrightarrow$ i of the active terminals transmit on channel j

$H_{i,j}(n) = H_i(n) \longleftrightarrow$ exactly i channels are occupied in addition to ch. j

$g_{i,j}(n) = g_i(n) := Pr(W_{i,j}(n))$; $g^{+}_{i,r,j}(n) = g^{+}_{i,r}(n) := Pr(W_{i,j}(n) \mid H_{r,j}(n))$

$c_{i,j} = c_i := Pr(T_{i,j})$; $h_{i,j}(n) = h_i(n) := Pr(H_{i,j}(n))$

We are now able to formulate the main theorem of this paper:

Theorem 4:

The conditional throughput of a (m,d,L)-ALOHA system is given by:

$$S_n(f) = \frac{m \cdot a_m \cdot (c_o \cdot n \cdot u_n \cdot (1-u_n)^{n-1} + c_1 \cdot (1-u_n)^n) \cdot}{(1 + \sum_{k=1}^{m-1} (\frac{a_k}{a_m} - 1) \cdot \binom{m-1}{k-1} \cdot c_o^{m-k} \cdot (1-c_o \cdot (1-u_n)^n)^{k-1} \cdot (1-u_n)^{n \cdot (m-k)})}$$

where

$u_n = f(n)/m := Pr(\text{a blocked terminal retransmits on ch. } j \mid n \text{ blocked term.})$

and a_k , c_i are as in theorems 1 - 3.

For n large, the optimal conditional throughput is obtained if

$u_n = b/n$ where b is a given constant.

Thus: $$S_n(f) \xrightarrow[n\to\infty]{} \frac{m \cdot a_m \cdot (c_o \cdot b + c_1) \cdot e^{-b} \cdot (1 + \sum_{k=1}^{m-1} (\frac{a_k}{a_m} - 1) \cdot \binom{m-1}{k-1} \cdot c_o^{m-k} \cdot}{(1-c_o \cdot e^{-b})^{k-1} \cdot e^{-b \cdot (m-k)})}$$

From this formula the optimal value of the parameter b may be derived by differentiation.

Proof: The conditional throughput is given by:

$$S_n(f) = \sum_{j=1}^{m} \sum_{k=1}^{m} \left(\Pr(W_{1,j}(n) \wedge T_{o,j} \wedge H_{k-1,j}(n)) + \Pr(W_{o,j}(n) \wedge T_{1,j} \wedge H_{k-1,j}(n))\right) \cdot a_k$$

$$= m \cdot \sum_{k=1}^{m} \left(\Pr(T_o) \cdot \Pr(W_1(n) \mid H_{k-1}(n)) \cdot \Pr(H_{k-1}(n) + \Pr(T_1) \cdot \Pr(W_o(n) \mid H_{k-1}(n)) \cdot \Pr(H_{k-1}(n)\right) \cdot a_k$$

$$= m \cdot \sum_{k=1}^{m} (c_o \cdot g^{+}_{1,k-1}(n) + c_1 \cdot g^{+}_{o,k-1}(n)) \cdot h_{k-1}(n) \cdot a_k$$

Furthermore:

$$g^{+}_{i,r,j}(n) = g^{+}_{i,r}(n) = \binom{n}{i} \cdot u_{r,n}^{\,i} \cdot (1 - u_{r,n})^{n-i}$$

where

$u_{r,n}$:= Pr(a blocked terminal retransmits over channel j | exactly r channels are occupied and there are n blocked stations)

$u_{r,n}$ is influenced by the number r of channels which are occupied (with retransmissions or new arrivals) in a given slot, but this dependency (if any) will disappear for larger values of n (the reader should observe that stability considerations are only necessary if the number n of blocked terminals is very large).

Thus in the following we restrict on

$$u_{r,n} := u_n = f(n)/m \ .$$

Remark: If we knew that none of the blocked terminals transmits over a channel which is different from j then we have the following upper bound for $u_{r,n}$:

$u_{r,n} \le u^{o}_{r,n}$:= Pr(a blocked terminal retransmits over channel j given that no blocked terminal retransmits over a channel j' ≠ j)

$$= \frac{u_n}{1 - (m-1)u_n} = u_n + O(u_n^2) \ .$$

It may be shown that choosing $u^{o}_{r,n}$ instead of u_n for the probabilities $u_{r,n}$ leads to the same observation (namely $u_n = O(1/n)$) for the optimal retransmission control policy.

In the same way, the probabilities $h_{k-1}(n)$ could depend on the number of blocked terminals which transmit over other channels, but this influence vanishes for large n by the same argument.

Thus $h_{k-1}(n)$ = Pr(k-1 channels are occupied in addition to channel j)

$$= \binom{m-1}{k-1} \cdot (1 - c_o \cdot g_o(n))^{k-1} \cdot (c_o \cdot g_o(n))^{m-k}$$

since Pr(channel j is idle) = $c_o \cdot g_o(n)$.

Defining $x := 1 - u_n$ we obtain:

$g_o(n) = (1-u_n)^n = x^n$ as well as $g_1(n) = n \cdot u_n \cdot (1-u_n)^{n-1} = n \cdot (1-x) \cdot x^{n-1}$.

$$\text{Thus: } S_n(f) = m \cdot (c_o \cdot g_1(n) + c_1 \cdot g_o(n)) \cdot \Big(\sum_{k=1}^{m-1} h_{k-1}(n) \cdot a_k + a_m \cdot (1 - \sum_{k=1}^{m-1} h_{k-1}(n)) \Big)$$

$$= m \cdot (c_o \cdot g_1(n) + c_1 \cdot g_o(n)) \cdot a_m \cdot \Big(1 + \sum_{k=1}^{m-1} (\frac{a_k}{a_m} - 1) \cdot h_{k-1}(n)\Big)$$

$$= m \cdot a_m \cdot (c_o \cdot n \cdot (1-x) \cdot x^{n-1} + c_1 \cdot x^n) \cdot (1 + \sum_{k=1}^{m-1} d_k \cdot (1-c_o \cdot x^n)^{k-1} \cdot (x^n)^{m-k})$$

where $d_k := (\frac{a_k}{a_m} - 1) \cdot \binom{m-1}{k-1} \cdot c_o^{m-k}$.

The partial derivative of $S_n(f)$ to the unknown x is easily shown to be:

$$\frac{\partial S_n(f)}{m \cdot a_{m-1} \cdot n^2 \cdot \partial x} = (c_o \cdot x^{n-2} - c_o \cdot x^{n-1}) \cdot (1 + \sum_{k=1}^{m-1} d_k \cdot (1-c_o \cdot x^n)^{k-1} \cdot (x^n)^{m-k})$$

$$+ (c_o \cdot x^{n-1} - c_o \cdot x^n) \cdot \sum_{k=1}^{m-1} d_k \cdot \big(-(k-1) c_o \cdot (1-c_o \cdot x^n)^{k-2} \cdot x^{n-1} \cdot (x^n)^{m-k}$$

$$+ (m-k) \cdot x^{n(m-k)-1} \cdot (1-c_o \cdot x^n)^{k-1}\big)$$

$$+ O(1/n)$$

$$= c_o \cdot (1-x) \cdot x^{n-2} \cdot (F(x) + x \cdot G(x)) + O(1/n).$$

Setting the partial derivative equal to zero we get insight into the asymptotic behaviour of the optimal values of x. The first solution is:

$$1 - x = O(1/n)$$

i.e. $u_n = 1 - x = O(1/n)$; $f(n) = u_n/m = O(1/n)$.

Thus for large n a retransmission control policy has been obtained which corresponds to the classical (1,1,*)-ALOHA system.

Remark: Another solution would be $x^{n-2} = O(1/n)$, i.e. $x \xrightarrow[n\to\infty]{} 1$, $u_n \xrightarrow[n\to\infty]{} 0$.

Thus: $S_n(f) = m \cdot a_m \cdot c_1 \cdot (1 + \sum_{k=1}^{m-1} (\frac{a_k}{a_m} - 1) \cdot \binom{m-1}{k-1} \cdot c_o^{m-k} \cdot (1-c_o)^{k-1})$

but this 'solution' cannot be accepted because a blocked terminal would never be active again; all of the throughput would be produced by the terminals which are still active (in the infinite user model it has been assumed that the rate of new arrivals doesnot depend on the number of blocked stations).

Setting $u_n := \frac{b}{n}$ the conditional throughput $S_n(f)$ is given by:

$$S_n(f) = m \cdot a_m \cdot (c_o \cdot b \cdot (1-\frac{b}{n})^{n-1} + c_1 \cdot (1-\frac{b}{n})^n) \cdot (1 + \sum_{k=1}^{m-1} d_k \cdot (1-c_o \cdot (1-\frac{b}{n})^n)^{k-1} \cdot \big[(1-\frac{b}{n})^n\big]^{m-k})$$

$$\xrightarrow[n\to\infty]{} m \cdot a_m \cdot (c_o \cdot b + c_1) \cdot e^{-b} \cdot (1 + \sum_{k=1}^{m-1} d_k \cdot (1-c_o \cdot e^{-b})^{k-1} \cdot e^{-b(m-k)}) .$$

The optimal value of the parameter b is determined by $\partial S_n(f)/\partial b \stackrel{!}{=} o$ leading to:

$$\frac{c_o}{y} \cdot \Big(1 + \sum_{k=1}^{m-1} d_k \cdot (1-c_o \cdot y)^{k-1} \cdot y^{m-k}\Big)$$

$$= (-c_o \cdot \ln(y) + c_1) \cdot \Big(1 + \sum_{k=1}^{m-1} d_k \big((k-1) \cdot (1-c_o y)^{k-2} \cdot (-c_o) \cdot y^{m-k+1} + (m-k+1) \cdot (1-c_o y)^{k-1} \cdot y^{m-k}\big)\Big)$$

where $y := e^{-b}$.

For $m > 1$ this equation can be numerically solved; an analytic expression for the general solution has not been found.

For $m = 1$ the above expression leads to:

$$b = \frac{c_o - c_1}{c_o} \quad , \quad \text{i.e.} \quad f(n) = u(n) = \frac{1 - c_1/c_o}{n} = f^*(n) + O(1/n^2)$$

where $f^*(n) = (1 - c_1/c_o)/(n - c_1/c_o)$ is the optimum retransmission probability derived in theorem 2.

V. THE DOMINATING CHANNEL MODEL (FM-ALOHA)

In some high frequency radio transmission systems it is possible to decode the strongest of several interfering signals (FM capture). Thus in this section we consider a model where the users are divided into m classes $K_1, \ldots, K_m$ $(m \geq 2)$ according to their emitter power levels $E_1 < E_2 < E_3 < \ldots < E_m$.

It will be assumed that channel K_i dominates over channel K_{i-1} $(i=2,\ldots,m)$, i.e. a packet of a K_i-user will be correctly received if and only if channels $K_{i+1}, \ldots, K_m$ are empty and there is no conflict with another transmission over K_i (cf. [M]). Like this it is possible to simulate a special version of a (m,1,o)-ALOHA system by only one instead of m user frequencies. If all of the classes are equally loaded then the throughput is given by theorem 3 since in this case the dominance of some channels over other channels results in longer waiting times for low power users, but has no effect on total throughput. It will be shown in the following that by means of unequal channel loads higher throughput rates can be obtained; this is no contradiction to theorems 3 and 4 since the probability that a packet is not accepted is now dependent on the number of the channel.

We assume that class i consists of a very large number of users $(i=1,\ldots,m)$ which produce a throughput rate $S(i)$ and a traffic rate $G(i)$ respectively, both of which are supposed to be Poisson processes. Furthermore there are no dependencies among the users.

Let $S_{max}^{(m)} := \max_{G(1),\ldots,G(m)} \Big\{ \sum_{i=1}^{m} S(i) \Big\}$ be the maximum obtainable throughput

of the system and $S(i)^+$ and $G(i)^+$ be the rates for which this maximum is obtained.

Thus: $S_{max}^{(m)} = S(1)^+ + S(2)^+ + \ldots + S(m)^+$.

Theorem 5 ([M]):

a. $S(i) = G(i) \cdot e^{-G(i)-G(i+1)-\ldots-G(m)}$

b. $S(r)^+ = G(r)^+ . e^{-G(r)^+-\ldots-G(m)^+}$

$$\sum_{i=1}^{r} S(i)^+ = e^{-G(r)^+-\ldots-G(m)^+}$$

where $G(r)^+$ is recursively defined by:

$$G(1)^+ = 1 \quad ; \quad G(j+1)^+ = 1 - e^{-G(j)^+} \quad (j=1,\ldots,m-1) .$$

c. $S_{max}^{(m)} = \sum_{i=1}^{m} S(i)^+ = e^{-G(m)^+}$.

The second part of this theorem is easily shown by induction: under the assumption that the optimal values $S(i+1)^+,\ldots,S(m)^+$ and $G(i+1)^+,\ldots,G(m)^+$ were already known the optimal values $S(i)^+$ and $G(i)^+$ are easily determined; following that, optimal values $S(i+1)^+$ and $G(i+1)^+$ are obtained (the reader should observe that $S(i+1)$ and $G(i+1)$ are not influenced by the traffic on channels $K_1,\ldots,K_i$).

Example:

Consider the dominating channel model with two power levels (m=2).

Then: $S_{max}^{(2)} = e^{-G(2)^+} = e^{-(1-e^{-1})} \approx 0.531$

$$G_{max}^{(2)} = G(1)^+ + G(2)^+ = 1 + (1-e^{-1}) \approx 1.632$$

[Remark:

For the (2,1,o)-ALOHA system the corresponding rates are:

$$S_{max} \approx o.522 \quad ; \quad G_{max} \approx 1.516$$

(these values are obtained from the relation $S(2,1,o) = \frac{G}{2} \cdot (e^{-G/2} + e^{-G})$).]

In the dominating channel model, the traffic rates and the probabilities of a successful transmission $p^{(i)} := S(i)^+/G(i)^+$ are given by:

$S(1)^+ \approx o.1955$; $G(1)^+ = 1$; $p^{(1)} \approx o.196$

$S(2)^+ \approx o.3359$; $G(2)^+ \approx o.63$; $p^{(2)} \approx o.531 = e \cdot p^{(1)}$

Thus the probability of collision (and, consequently, the mean waiting time) is substantially lower for class K_2-users than for K_1-users.

For the dominating channel model with m channels the following relations are obtained (cf. [Sp2]):

a. $p^{(r+1)}/p^{(r)} = e^{G(r)^+} > 1 \qquad (r = 1,\ldots,m-1)$.

b. $$e = \frac{p^{(2)}}{p^{(1)}} > \frac{p^{(3)}}{p^{(2)}} > \ldots\ldots > \frac{p^{(m)}}{p^{(m-1)}} \xrightarrow[m\to\infty]{} 1$$

$$e > \frac{T^{(1)}}{T^{(2)}} > \frac{T^{(2)}}{T^{(3)}} > \ldots\ldots > \frac{T^{(m-1)}}{T^{(m)}} \xrightarrow[m\to\infty]{} 1$$

where $T^{(i)}$ is the mean waiting time of a channel K_i-user.

The following result describes the asymptotic behaviour of $S(i)^+$ and $G(i)^+$:

Theorem 6 ([Sp1]):

a. $G(i)^+ > G(i+1)^+$

b. $\frac{1}{i} \le G(i)^+ < \frac{2}{i}$ $\Big\}$ $(i = 1,\ldots,m)$

c. If $\frac{1+\varepsilon}{i} \le G(i)^+$ where $0 \le \varepsilon < 1$ and $\frac{\varepsilon(1+\varepsilon)}{1-\varepsilon} \le i$, then

$$\frac{1+\varepsilon}{r} \le G(r)^+ < \frac{2}{r} \quad \text{for all} \quad r \ge i \ .$$

d. $$S(1)^+ + \ldots + S(m)^+ \xrightarrow[m\to\infty]{} 1$$
$$G(1)^+ + \ldots + G(m)^+ \xrightarrow[m\to\infty]{} \infty$$

The same result is obtained from theorem 3 for (m,d,L)-ALOHA systems.

In figure 5 the dominating channel model has been compared with the (m,1,o)-model. The dominating channel values are generally higher but the optimal throughput rates differ by less than o.o3.

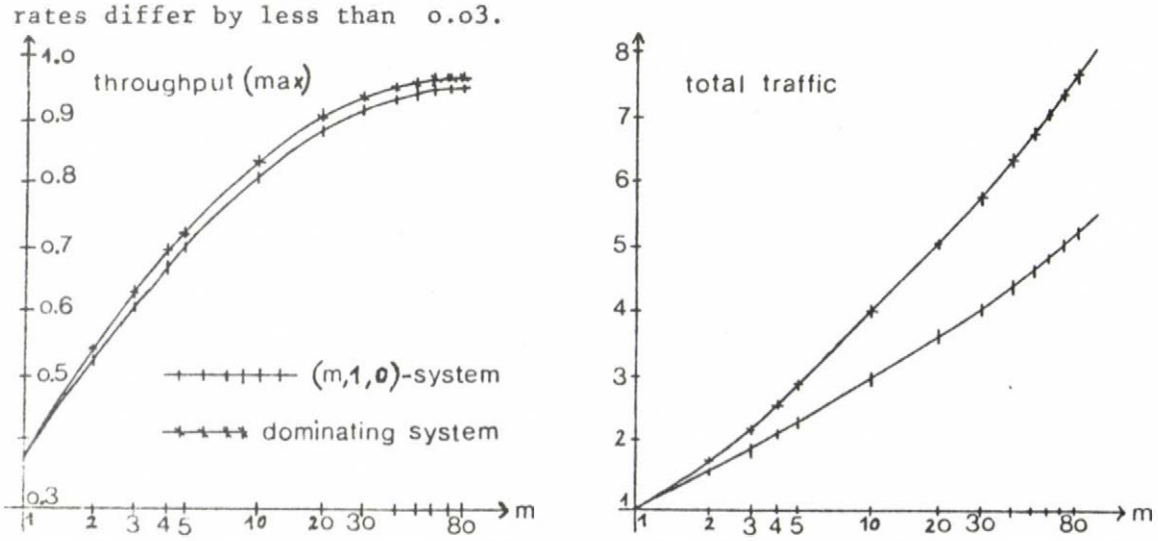


FIGURE 5

VI. STABILITY DISCUSSION OF THE DOMINATING CHANNEL MODEL

The results of sections II and IV concerning the stabilization of modified ALOHA systems may be generalized on the dominating channel model which has been introduced in the preceding section. It turns out that the optimum retransmission control probability for channel K_r-users depends on the number n_j of blocked terminals of class j ($j = 1,\ldots,r$) as well as on the rates of new arrivals to these classes; furthermore the retransmission probability f_r of class r is inversely proportional to the number of blocked terminals which belong to class r; compared to the corresponding result of theorem 2, f_r is reduced by the number of blocked stations in lower classes (this results in a higher success probability for packets which originate from low power stations).

Theorem 7:

Let n_i := the actual number of blocked terminals which belong to class i

$c_k^{(i)}$:= Pr(exactly k new arrivals originating from channel K_i)

$g_k^{(i)}(n_i)$:= Pr(exactly k retransmissions on channel K_i | n_i blocked in class i)

$$= \binom{n_i}{k} \cdot f(n_i)^k \cdot (1-f(n_i))^{n_i-k} .$$

The optimum retransmission probability f_r for channel K_r-users is given by:

$$f_r = \Pr(\text{a blocked terminal in class r retransmits} \mid (n_1,\ldots,n_m) \text{ blocked in classes } 1,\ldots,m)$$

$$= f_r(n_r,\lambda_r,n_{r-1},\lambda_{r-1},\ldots,n_1,\lambda_1)$$

$$= \frac{1 - E_r - \lambda_r}{n_r - E_r - \lambda_r} = O(1/n_r)$$

where: $\lambda_r = c_1^{(r)}/c_o^{(r)}$

$$E_r = \sum_{i=1}^{r-1} D_{n_i}^{(i)} \cdot \prod_{j=i+1}^{r-1} c_o^{(j)} \cdot (1-f_j)^{n_j}$$

= conditional throughput of classes $(1,\ldots,r-1)$ given that higher classes are inactive.

and $D_{n_i}^{(i)} = c_o^{(i)} \cdot g_1^{(i)}(n_i) + c_1^{(i)} \cdot g_o^{(i)}(n_i)$

$$= c_o^{(i)} \cdot \left(\left(\frac{n_i - 1}{n_i - E_i - \lambda_i} \right)^{n_i - 1} - \left(\frac{n_i - 1}{n_i - E_i - \lambda_i} \right)^{n_i} \cdot E_i \right)$$

= conditional throughput of class i given that higher classes are inactive.

By means of these formulae f_r may be expressed in terms of n_j and $c_i^{(j)}$ ($j \in \{1,\ldots,r\}$, $i \in \{o,1\}$).

Proof:

Since higher channel transmissions dominate over packets which come from lower powered stations, the total conditional throughput is given by:

$$
\begin{aligned}
& S_{(n_1,\ldots,n_m)}(f_1,\ldots,f_m) \\
&= \sum_{i=1}^{m} \left((c_o^{(i)} \cdot g_1^{(i)}(n_i) + c_1^{(i)} \cdot g_o^{(i)}(n_i)) \cdot \prod_{j=i+1}^{m} c_o^{(j)} \cdot g_o^{(j)}(n_j) \right) \\
&= \sum_{i=1}^{m} \left((c_o^{(i)} \cdot n_i \cdot f_i \cdot (1-f_i)^{n_i-1} + c_1^{(i)} \cdot (1-f_i)^{n_i}) \cdot \prod_{j=i+1}^{m} c_o^{(j)} \cdot (1-f_j)^{n_j} \right) \\
&= \sum_{i=1}^{m} \left((c_o^{(i)} \cdot n_i \cdot x_i^{n_i-1} + (c_1^{(i)} - n_i \cdot c_o^{(i)}) \cdot x_i^{n_i}) \cdot \prod_{j=i+1}^{m} c_o^{(j)} \cdot x_j^{n_j} \right)
\end{aligned}
$$

where $x_i := 1 - f_i$.

The partial derivative of S with respect to x_r is given by:

$$
\begin{aligned}
\frac{\partial S}{\partial x_r} = & \left((n_r - 1) \cdot n_r \cdot c_o^{(r)} \cdot x_r^{n_r-2} + (c_1^{(r)} - n_r \cdot c_o^{(r)}) \cdot n_r \cdot x_r^{n_r-1} \right) \cdot \prod_{j=r+1}^{m} c_o^{(j)} \cdot x_j^{n_j} \\
& + \sum_{i=1}^{r-1} \left(\frac{n_r}{x_r} \cdot (c_o^{(i)} \cdot n_i \cdot x_i^{n_i-1} + (c_1^{(i)} - n_i \cdot c_o^{(i)}) \cdot x_i^{n_i} \right) \cdot \prod_{j=i+1}^{m} c_o^{(j)} \cdot x_j^{n_j}
\end{aligned}
$$

In order to determine the maximum throughput we have to evaluate the equations

$$\frac{\partial S}{\partial x_r} = 0 \quad \text{for } r = 1,\ldots,m .$$

By means of $\frac{\partial S}{\partial x_1} \stackrel{!}{=} 0$ the optimal values of f_1 and n_1 are obtained:

$$
\begin{aligned}
\frac{\partial S}{\partial x_1} = 0 \quad \Rightarrow \quad & \left((n_1 - 1) \cdot n_1 \cdot c_o^{(1)} \cdot x_1^{n_1-2} + (c_1^{(1)} - n_1 \cdot c_o^{(1)}) \cdot \overbrace{n_1 \cdot x_1^{n_1-1} \cdot \prod_{j=2}^{m} c_o^{(j)} \cdot x_j^{n_j}}^{} \right) = 0 \\
\Rightarrow \quad & x_1 = \frac{(n_1 - 1) \cdot c_o^{(1)}}{n_1 \cdot c_o^{(1)} - c_1^{(1)}} \\
\Rightarrow \quad & f_1 = 1 - x_1 = \frac{c_o^{(1)} - c_1^{(1)}}{n_1 c_o^{(1)} - c_1^{(1)}} = \frac{1 - \lambda_1}{n_1 - \lambda_1} \quad \text{where } \lambda_1 = \frac{c_1^{(1)}}{c_o^{(1)}} .
\end{aligned}
$$

The optimal values of f_2 and n_2 (and so on) may be inductively determined. The general step is as follows:

Assume that $f_1,\ldots,f_{r-1};x_1,\ldots,x_{r-1};D_{n_1}^{(1)},\ldots,D_{n_{r-1}}^{(r-1)}$ are already known and that $f_1,\ldots,f_{r-1}$ may be expressed in terms of $x_1,\ldots,x_{r-1}$ and of the new arrival rates. Then $\partial S/\partial x_r$ may be expressed in the following form:

$$\frac{\partial S}{\partial x_r} = \left((n_r-1)\cdot n_r . c_o^{(r)} \cdot x_r^{\,n_r-2} + (c_1^{(r)} - n_r \cdot c_o^{(r)}) \cdot n_r \cdot x_r^{\,n_r-1}\right) \cdot \prod_{j=r+1}^{m} c_o^{(j)} \cdot x_j^{\,n_j}$$

$$+ \sum_{i=1}^{r-1} \frac{n_r}{x_r} \cdot D_{n_i}^{(i)} \cdot \prod_{j=i+1}^{m} c_o^{(j)} \cdot x_j^{\,n_j} \stackrel{!}{=} o \; .$$

Thus: $$(n_r-1)\cdot c_o^{(r)} \cdot x_r^{\,n_r-2} + (c_1^{(r)} - n_r \cdot c_o^{(r)}) \cdot x_r^{\,n_{r-1}} + \sum_{i=1}^{r-1} \frac{D_{n_i}^{(i)}}{x_r} \cdot \prod_{j=i+1}^{r} c_o^{(j)} \cdot x_j^{\,n_j} = o$$

$$\Rightarrow \quad (n_r-1) + (\lambda_r - n_r)\cdot x_r + \underbrace{\left(\sum_{i=1}^{r-1} D_{n_i}^{(i)} \cdot \prod_{j=i+1}^{r-1} c_o^{(j)} \cdot x_j^{\,n_j} \right)}_{E_r} \cdot x_r = o$$

$$\Rightarrow \quad x_r = \frac{n_r - 1}{n_r - \lambda_r - E_r} \quad \Rightarrow \quad f_r = 1 - x_r = \frac{1 - E_r - \lambda_r}{n_r - E_r - \lambda_r} \; .$$

For $D_{n_r}^{(r)}$ the following expression is obtained:

$$D_{n_r}^{(r)} = c_o^{(r)} \cdot g_1^{(r)}(n_r) + c_1^{(r)} \cdot g_o^{(r)}(n_r)$$

$$= c_o^{(r)} \cdot n_r \cdot (1-f_r)^{n_r-1} + c_1^{(r)} \cdot (1-f_r)^{n_r}$$

$$= (1-f_r)^{n_r-1} \cdot \left(c_o^{(r)} \cdot n_r \cdot f_r + c_1^{(r)} \cdot (1-f_r) \right)$$

$$= (1-f_r)^{n_r-1} \cdot \left(c_o^{(r)} \cdot n_r \cdot (1-E_r-\lambda_r) + c_1^{(r)} \cdot (n_r-1) \right) / (n_r-E_r-\lambda_r)$$

$$= (1-f_r)^{n_r-1} \cdot c_o^{(r)} \cdot \left((n_r - n_r \cdot E_r - \frac{c_1^{(r)}}{c_o^{(r)}}) / (n_r-E_r-\lambda_r) \right)$$

$$= (1-f_r)^{n_r-1} \cdot c_o^{(r)} \cdot \left(1 - \frac{(n_r-1)\cdot E_r}{n_r-E_r-\lambda_r} \right)$$

$$= c_o^{(r)} \cdot \left(\frac{n_r - 1}{n_r-E_r-\lambda_r} \right)^{n_r-1} \cdot \left(1 - \frac{(n_r-1)\cdot E_r}{n_r-E_r-\lambda_r} \right) \; .$$

This completes the proof of theorem 7.

REFERENCES

[A] Abramson, N. The Throughput of Packet Broadcasting Channels. IEEE Trans. on Comm., vol. COM-25, (1977), 117-128. (This paper contains a very comprehensive list of papers which have been published in this field)

[FGL] Fayolle, G. Gelenbe, E. Labetoulle, L. Stability and Control of Packet-Switching Broadcast Channels. Journal of the ACM, vol. 24,(1977), 375-386.

[KL] Kleinrock, L. Lam, S. Packet Switching in a Multiaccess Broadcast Channel. IEEE COM-23, (1975), 410-422 and 891-904.

[M] Metzner, J.J. On Improving Utilization in ALOHA Networks. IEEE COM-24, (1976), 447-448.

[S1] Spaniol, O. Analyse von ALOHA-Netzwerken mit mehreren Benutzerkanälen. Universität Bonn, Informatik-Bericht Nr. 12 (1976).

[S2] Spaniol, O. Mehrfrequenz-ALOHA-Netzwerke. Informatik-Fachberichte Nr. 9, Springer-Verlag (1976), 277-295.

[T] Tobagi, F. Kleinrock, L. Packet Switching in Radio Channels: Parts I - III. IEEE COM-23, (1975), 1400-1433; IEEE COM-24, (1976), 832-844.

SYMBOLS

N	number of user terminals
m	" of user frequencies
d	" of retransmission frequencies
L	maximum queue length
L^+	actual queue length
$S = S(m,d,L)$	throughput of a (m,d.L)-ALOHA network
$G = G(m,d,L)$	total traffic
p_k	Pr(k transmissions in a given slot)
$g_k(n)$	Pr(k retransmissions | n blocked terminals)
c_k	Pr(k new transmissions)
a_k	Pr(information arriving on channel j is accepted | k channels occ.)
q_k	Pr(k channels are occupied)
$f(n)$	Pr(a blocked station retransmits | n blocked)
$S_n(f)$	conditional throughput given that there are n blocked terminals
$S(f)$	unconditional throughput of a system controlled by the retransmission control policy f(n)
$u_n = f(n)/m$	Pr(a blocked terminal retransmits on channel j | n blocked)
$S(i)$	throughput of class i (dominating channel model)
$G(i)$	traffic of class i
$S(i)^+$	optimal throughput of class i
$G(i)^+$	" traffic " " i
$p(i)$	probability of successful transmission in class i
$S_{max}^{(m)}$	optimal troughput (dominating channel model)
n_i	number of blocked terminals in class i
$c_k^{(i)}$	Pr(k new arrivals in class i)
$g_k^{(i)}(n_i)$	Pr(k retransmissions in class i | n_i blocked in class i)
E_r	conditional throughput of classes (1,...,r-1) given that higher channels are inactive
$D_{n_i}^{(i)}$	conditional throughput of class i given that higher channels are inactive
f_r	Pr(a blocked terminal in class r retransmits | $(n_1,...,n_m)$ blocked in classes 1,...,m)
$T^{(i)}$	mean waiting time of a class i packet
K_i	channel (class, frequency) i
$\lambda = c_1/c_o$	
$\lambda_r = c_1^{(r)}/c_o^{(r)}$	
$x_r = 1 - f_r$	

ON THE BASIC CONCEPTS OF A MODULE LANGUAGE

G. DÁVID

Introduction

The theory of Petri nets provides an effective tool to describe control structures. Termination, deadlooks can be expressed by this method. In this paper we give an extension of the notion of Petri nets as a programming language by the definition of abstract data types, abstract control types and a verification-method will be treated also.

Kotov [5] gives a method to formalize and translate Petri nets into the expressions of a formal language and introduces the notion of "abstract control structures" ACS. Dávid and others [1, 2] described a special logic, the Structure Logic which allows to express statements on "abstract data structures" ADS. This logic is a programming language, too, a new-principled language like PROLOG i.e. a language without algorithmization. Here we want to combine these two results in the notion of the module. A module consists of two sets of formal parameters: abstract control parameters and data parameters:

module name (formal ACS; formal ADS)

We call this language Module Language ML.

Main goal of this paper is to show how one can transform Petri nets and data into programs in a formal language ML or more generally, the

(ACS, ADS) ⟷ programs in ML

equivalence will be treated which is similar to the everydays equivalence between the flowcharts and programs:

flowcharts ⟷ programs

in programming languages.

In the first two chapters we give a short description of the ACS and ADS, in the third the module-language ML and verification method in ML will be treated.

1. Abstract control structures ACS

Here we give a short overview of this notion, details are available in Kotov [5]. We define

- meta-operations over nets
- operations over nets
- abstract control structures.

A basic symbol represents an atomic Petri net

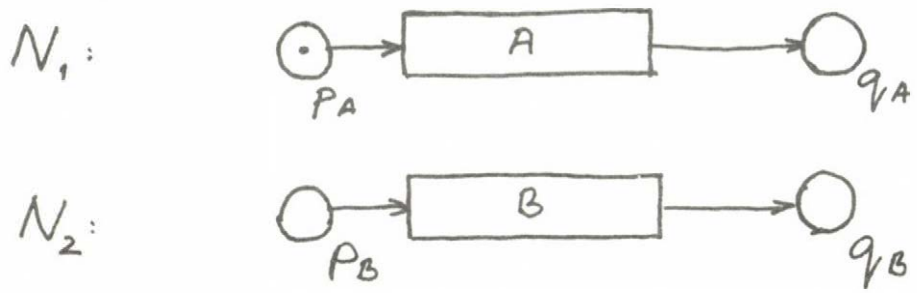
where
p_A - head place
q_A - tail place
A - transition
$M(N_1)$ - number of tokens

Here A represents a Petri net (compound transition) or a symbol (simple transition). Firing the compound transition is equivalent with the initialization of the nested Petri net and

when the internal net stops, the compound transition terminates and the initial marking of the nested Petri net is restored.

1.1. Meta-operations: union, elementary merge and merge

Let be given two atomic Petri nets N_1 and N_2

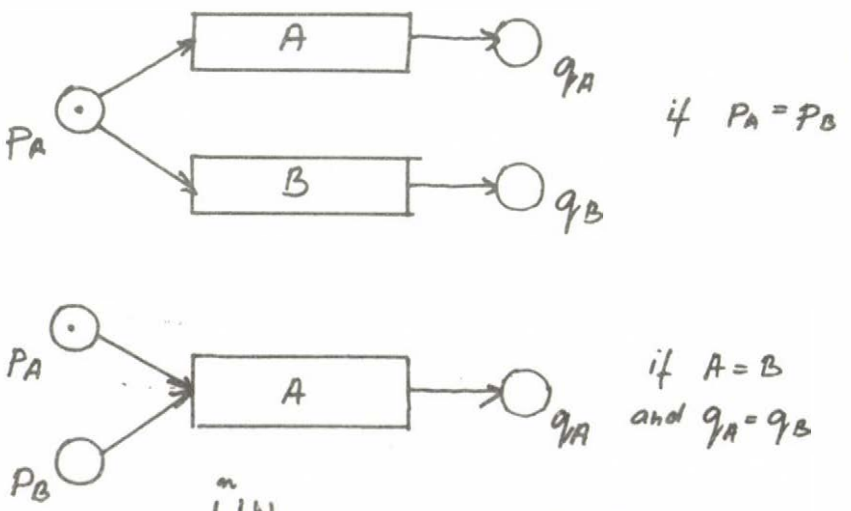


U union:

$$N_1 \cup N_2 = (P_A \cup P_B, A \cup B, q_A \cup q_B)$$

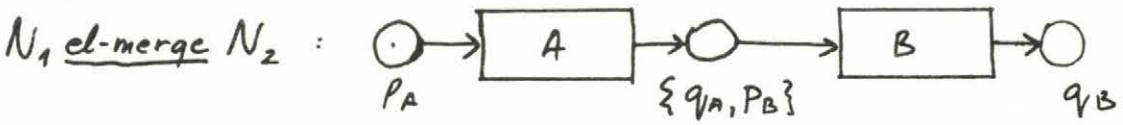
where $\cup$ represents the set-theoretical union. $M(N_1 \cup N_2)$ is defined as originally if $p_A \neq p_B$ and max. $(M(N_1), M(N_2))$ if $p_A = p_B$.

Examples:



Similarly one can define $\bigcup_{i=1}^{n} N_i$ for a set of atomic nets $\{N_1, N_2, \ldots, N_n\}$

el-merge, elementary merge meta-operation /not defined by Kotov/



i.e. the tail place q_A and the head place p_B of N_1 and N_2 resp. form a new place $\{q_A, p_B\}$ (ordered pair).
If the Petri nets $\mathcal{N}_1$ and $\mathcal{N}_2$ are compound ones and

$$\mathcal{N}_1 = \bigcup_{i=1}^{n_1} N_1^i \;, \quad \mathcal{N}_2 = \bigcup_{j=1}^{n_2} N_2^j \quad \text{then}$$

merge meta operation is defined as

$$\mathcal{N}_1 \text{ merge } \mathcal{N}_2 = \bigcup_{i,j}^{n_1,n_2} N_1^i \text{ el-merge } N_2^j$$

for every pair (i,j) , $i = 1,\dots,n_1$, $j = 1,\dots,n_2$.

1.2. Operations over nets

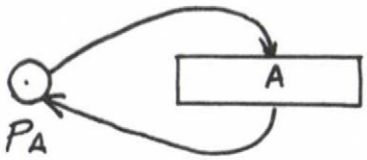
Using meta-operations, the following operations are defined

superposition	,	binary, associative, commutative
join	;	
exclusion	[]	
iteration	*	monadic
marking	$n \rightarrow$	monadic
closure	do ... od	monadic

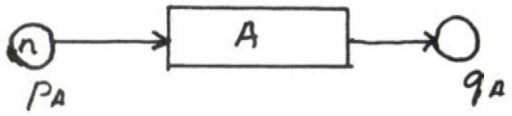
The operation superposition is defined as the union $\cup$, the operation join as merge.

For the monadic operations:

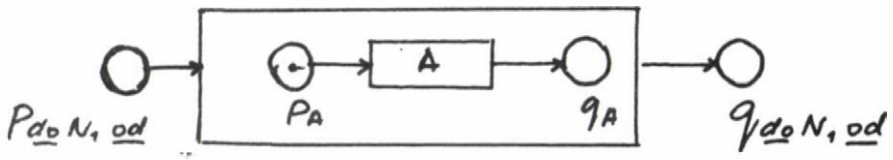
iteration: $*N_1$



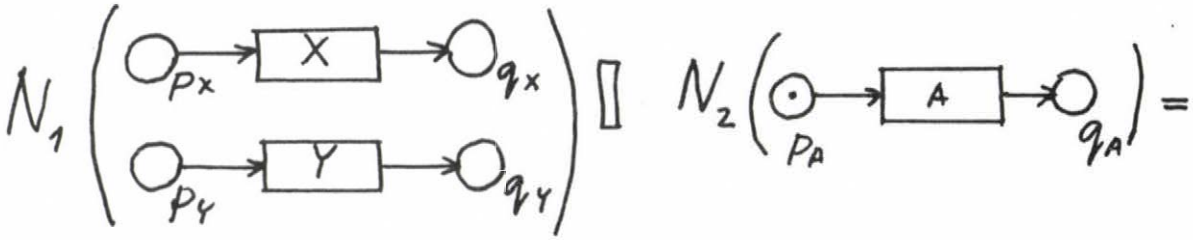
marking: $n \rightarrow N_1$

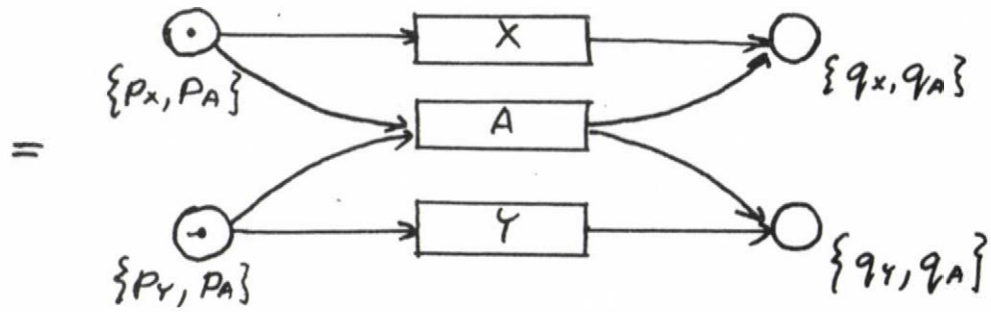


closure: do N_1 od



In order to describe the operation exclusion [] , first we give an example, introduce a new meta operation el-ex: elementary exclusion for two atomic nets and we shall extend this for compound nets:
An example of the binary operation exclusion



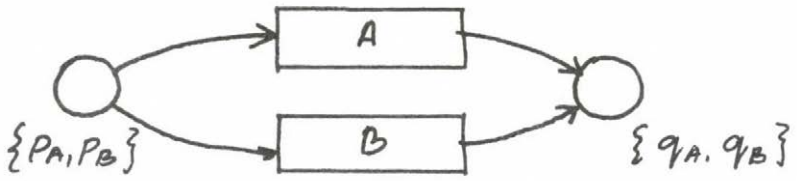


We see that $N_1 \,[\!]\, N_2$ exclusively uses the resources represented by $\{p_X, p_A\}$ and $\{p_Y, p_A\}$.

If N_1 and N_2 represent atomic nets as in the section 1.1, then

el-ex elementary exclusion:

N_1 el-ex N_2 :



where the ordered pairs $\{p_A, p_B\}$ and $\{q_A, q_B\}$ represent new head and tail places, resp.

If $\mathcal{N}_1$ and $\mathcal{N}_2$ are the same as in section 1.1, then

the binary exclusion operation $[\!]$:

$$\mathcal{N}_1 \,[\!]\, \mathcal{N}_2 = \bigcup_{i,j}^{n_1,n_2} N_1^i \text{ el-ex } N_2^j$$

Here we assume that N_1^i or N_2^j can be compound nets, using the operations el-merge or merge but not union of more elementary nets.

Kotov in [5] had not defined the meta operation el-merge and el-ex, but we introduced them for the simplicity's sake. Algebraic behaviour of operations (commutativity, binding, etc.) is discussed in [5].

Well-formed formulas wff's

⟨wff⟩ ::= ⟨operand⟩ | ⟨monadic operation⟩(⟨wff⟩) |
(⟨wff⟩⟨binary operation⟩⟨wff⟩)
⟨operand⟩ ::= ⟨atomic net symbol⟩

Examples:

$$\alpha \longrightarrow (A ; B)$$
$$1 \longrightarrow *((a \,[\!]\, b) \,[\!]\, c)$$

where A, B, a, b, c are atomic net symbols.
We shall omit some superfolous parentheses.

1.3. ACS: Abstract control structures are declared as

type control structure-name (formal parameters) = formula

where "formula" - is a well formed formula, using the symbols in the list of "formal parameters" as operands and optionally other ACS's, resulting an (atomic) Petri net.

In [5] two types of parameters are distinguished: the static ones (the symbols) and the dynamic parameters which are declared with the symbol preceeded by the basic word var. Hence var α in the list of formal parameters represents a set of Petri nets. It should be noted that in [5] ACS is called "abstract control types". Actual parameters for static ones are standard, for dynamic parameters we use the form [set of actual elements], i.e. the list of actual elements form a set, bracketed by [...], and this stands for a dynamic parameter.

Standard ACS's:

$$\begin{aligned} \text{join}\,(\text{var } x) &= 1 \longrightarrow (x_1 ; x_2 ; \cdots) \\ \text{exc}\,(\text{var } x) &= 1 \longrightarrow (x_1 \,[\!]\, x_2 \,[\!]\, \cdots) \qquad x_i \in X \\ \text{com}\,(\text{var } x) &= 1 \longrightarrow (x_1, x_2, \cdots) \end{aligned}$$

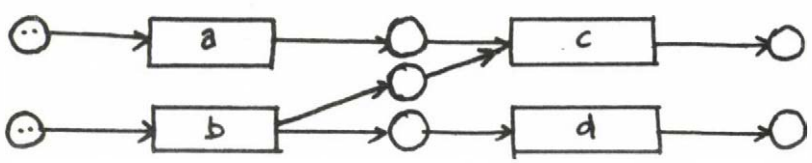
Hence

$$\text{join}\,([a, b, c]) = 1 \longrightarrow (a ; b ; c)$$

is an actual "call" of the join standard abstract control structure.

Examples

```
type  bridge (a,b,c,d)  =  2→(a;c,  b;d,  b;c)
```




```
type  loop  (var x)    =  1→*join (var x)
type  mutex (var x)    =  1→*exc  (var x)
type  vol   (a,b)      =  n→(a,b)
```

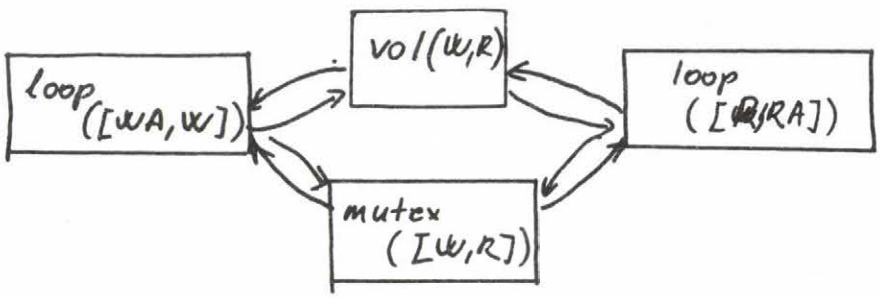
An example of a program of Reader/Writer problem using the examples above :

let be given
R:reader
W:writer
RA:action of reader R
WA:action of writer W

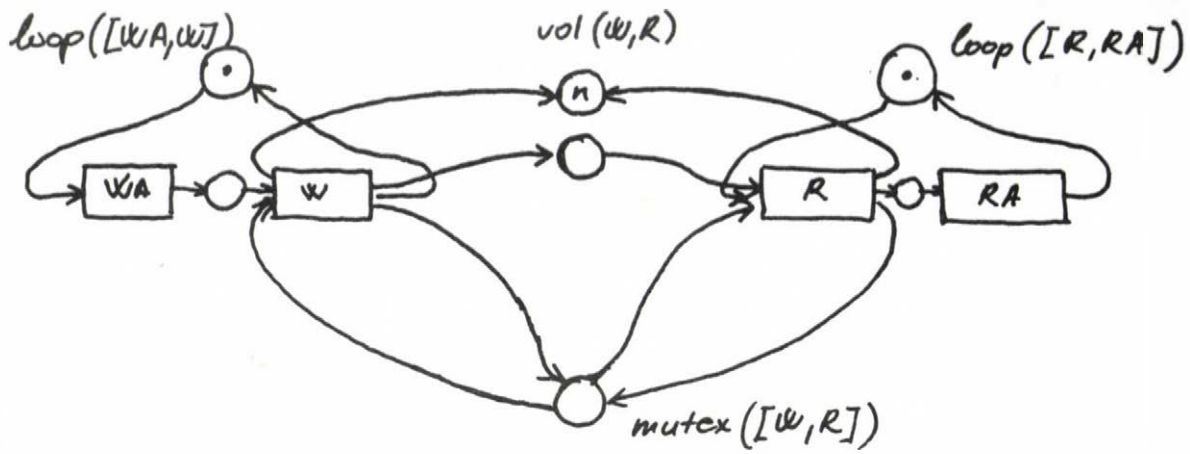
The program (as in [5])

```
do    loop ([WA, W]) ,
      loop ([R, RA]) ,
      mutex ([W,R]) ,
      vol  (W,R)                od
```

or in graphic form with comments :



or detailed :



2. Abstract data structures ADS

In this chapter we give a short introduction to SL, the Structure Logic [1, 2, 3, 4], in which one may define ADS and formalize assertions. SL is a logic, based on many-sorted logic extended by the notion of selectors. Hence SL as programming language has correct semantics /i.e. the logic itself/.

2.1. The Structure Logic SL

Let $\mathcal{T}$ be a given set of basic types /for example, integer, real, boolean etc./. Let us construct a composed type t from the basic types $t_i \in \mathcal{T}$ $i=1,2,\dots,n$ with the following sentence of this language

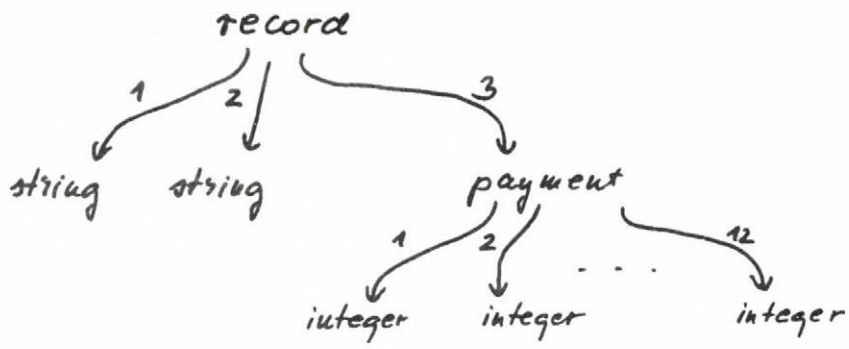
$$t < \{ sel_i : t_i \} >$$

$(\{ \dots \})$ stands for set , where symbols sel_i are arbitrary but different symbols, representing selectors, and sel_i selects a more elementary type t_i .

$t[sel_i]$ represents also the ith elementary type t_i :

$$t[sel_i] = t_i$$

Example



record ⟨1:string, 2:string, 3:payment⟩
payment ⟨1:integer, 2:integer, ..., 12:integer⟩

The type payment can be abbreviated as

payment ⟨[1:12] :integer⟩

and this is a set of statements payment [i] = integer, i = 1, ..., 12. These are <u>homogeneous</u> types. From composed or basic types one can similar define new composed types. A data-object can be described by this formalizm also

z⟨1:SMITH, 2:J.S., 3:⟨1:1251, 2:1742, ..., 12:1147⟩⟩
and $z[3][2]$ = 1742. A statement like

record⟨x⟩

can be valuated as <u>true</u> or <u>false</u>: true iff the object x satisties the requirements expressed by "record", and false if it doesn't. The sentences of this language are the formulas, consisting of atomic formulas $t\langle x\rangle$, where t - a type, x - an object and using the logical connectives $\wedge, \vee, \Rightarrow, \neg$ (not).

For more detailed introduction see [1, 2, 3, 4].

2.2. Declaration of Abstract Data Structures

For our goals we should declare abstract data types and data structures.

The declaration

type data-type-name $\langle sel_1:t_1, sel_2:t_2, \ldots, sel_n:t_n \rangle$

where for t_i $i = 1, \ldots, n$ either $t_i \in \mathcal{T}$ or t_i has been declared elsewhere as "data-type-name", and symbols sel_i $i = 1, 2, \ldots, n$ are different ones

Examples /as above/

type record $\langle$ 1:string, 2:string, 3:payment$\rangle$;
type payment $\langle$[1:12]:integer$\rangle$;

It should be noted that "data-type-name" must be unique - no redeclaration is allowed. The declared data-type-name is called abstract data structure ADS. Objects are declared by reference (or by ref)

reference (data-type-name) $data_1, data_2, \ldots, data_k$;

by which the symbol $data_j$ is declared as it is of type ADS sepcified by "data-type-name".

The language allows the combined form, by which in an ADS declaration one can use data objects as well as subtypes if the data object has been declared elsewhere:

type pay $\langle$1: NAME, 2:string, 3:payment$\rangle$;

if the NAME:

reference (string) NAME;

In that case the "pay" represents that set of data, in which the first components always the same data-object NAME: if

ref (pay) D1, D2;

then

D1 [1] = D2 [1], /i.e. the NAME/

and the remaining component may be different. By this data-synchronized processes may be specified, because D1 [1] varies as D2 [1]. We may also see that the statement

pay⟨x⟩ ⇒ record⟨x⟩

is a valid formula in Structure Logic.

3. Modules and verification

3.1. Modules

The Module Language ML is a language in which ACS and ADS can be described and the basic unit, the module gives the correspondance between control and data structures.

Formally a module is declared as:

```
module  module-name (formal abstract control structures;
                     formal abstract data structures)
        specification-part;
do      body od;
```

The "module-name" should be unique. A module has two lists of formal parameters: the list of formal abstract control

structures ACS, and the list of formal abstract data structures, separated by semicolon.

The specification-part describes the formal parameters.

General form

reference (set of allowed abstract structures) list of formal parameters;

where "list of formal parameters" contains those parameters f_i $i = 1, \ldots, k_1$, which are specified by a set of allowed structures s_j, $j=1,2,\ldots,n_2$, and it means that for arbitrary i

$$f_i = s_1 \quad \vee \quad f_i = s_2 \quad \vee \ldots \vee \quad f_i = s_{n_1}$$

i.e.

for $S = \{ s_i \mid i = 1, \ldots, n_1 \}$

f_i is of type s_j for some j, $s_j \in S$.

For simplicity we instroduce the set:

set set-name = list of set-elements;

for example

set execution = mutex, parex, seqex;

represents a set of different executions: mutual exclusion mutex, parallel execution parex, sequential execution seqex:

type mutex (var x) = 1 ⟶ exc (var x)
type seqex (var x) = 1 ⟶ join(var x)
type parex (var x) = 1 ⟶ com (var x)

The set declared should be homogeneous: only ADS's or only ACS's can be listed as set-elements in a declaration. Set

operations are: $\cup$ union, $\cap$ intersection, $-$ negation.

If integer, exp and mant are basic data types, and "real" is defined as

type real⟨1:exp, 2:mant⟩

and

set number = integer, real;

then

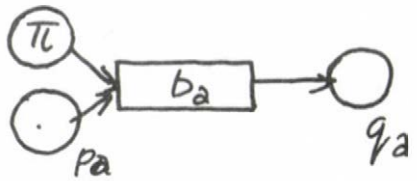
module whoknows (contr; z)
reference (execution) contr;
reference (number) z;

is a correct declaration and specification part declaring contr as one of the possible execution types and z is either integer or real. As in SL, a statement like integer⟨z⟩ may be true or false, and using logical connectives $\wedge$ /and/, $\vee$ /or/, $\neg$ /not/, $\Rightarrow$ /implies/ one can form quantifier-free well-formed formulas wff's in SL. For uniformity we shall use $t(x)$ instead of $t\langle x\rangle$ to express the fact that x is of type t.

In the module-body the programmer defines net-elements /or simply elements/ and the control-structure either by formal parameters or by another ACS already declared. Net-elements are local to the module, and the control structure is defined on net-elements in the module. Elements are Petri nets in a slightly modified form: a net-element a

$$a: \quad \pi \rightarrow b_a$$

where π - well formed formula in SL and b_a either a called module or (compound or simple) transition. The b_a will be executed, if a will be and π is true, and b_a acts as no-operation if π is false:

a : 

Simple transitions are the statements

$$x := f(y)$$

from a defined set of functions in the implementation and expressions on data objects x, y, e.q.

```
do  x[1] := log(x),
    x[2] := x/2**log(x)  od ;
```

Compound transitions are compound Petri nets. Also in the module--body one should specify the control-structure operating on the net-elements, where one can use either formal ACS-parameters as well as other ACS.

Using the examples above:

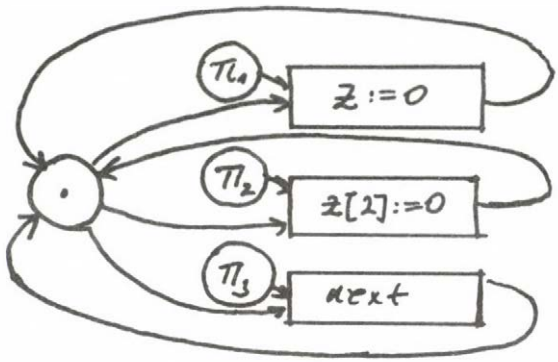
```
module  whoknows (contr; z);
ref  (execution)  contr; ref  (number) z;
do  a: integer (z)     → z:=0 ;
    b: real (z)        → z[2]:=0 ;
    c: ¬ integer (z) ∧ ¬ real(z) → next (parex; z);
    contr ([a,b,c]) ;
od;
```

where next represents a call of another module next. Three possible executions of this module when called:

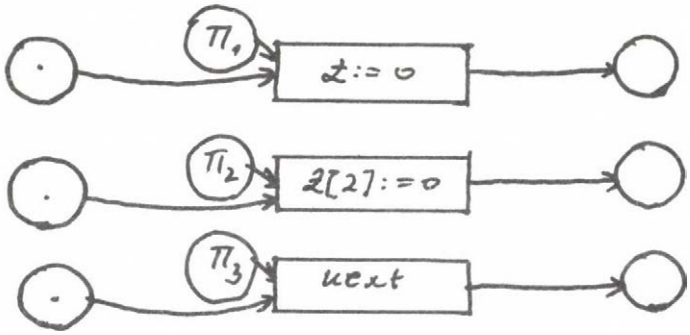
```
i/     whoknows (mutex, z)
ii/    whoknows (parex, z)
iii/   whoknows (seqex, z)
```

If π_1 = integer(z), π_2 = real(z), $\pi_3 = \neg\pi_1 \wedge \neg\pi_2$, then

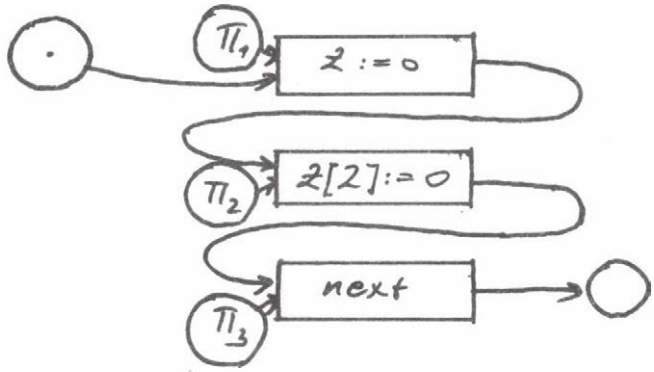
i)



ii)



iii)



In this language a program consists of a set type--declaration (i.e. for both abstract control and data structures), a set of module-declarations. These sets should be closed in that sense that every symbol could be derived from elementary symbols (elementary Petri-nets, basic data-types or modules).

For simplicity we shall assume that the ADS´s ACS´s, sets are declared spearately and they are global symbols, preceeded

by the word program:

```
program  program-name:
         set of set-declarations;
         set of type-declarations;
         set of module-declarations;
         program-name: wff
```

where wff is a well-formed formula over the modules as compound Petri-nets with actual parameters. (Logically the program and the module should not be different and distinguished in this form. We introduced this notion of program here to avoid discussion on local and global declarations etc.)

3.2. Verification

A user may insert assertions into his programs and may verify the correctness of the program. Two types of deductions are necessary: deductions in SL, and deductions of statements on abstract control structures.

3.2.1. Deductions in SL

Let $\mathcal{P}$ an $\mathcal{Q}$ be assertions expressed in SL. In SL two basic deductions exist:

1. i/ modus ponens $(\mathcal{P} \wedge (\mathcal{P} \Rightarrow \mathcal{Q})) \Rightarrow \mathcal{Q}$

1. ii/ $\left(\bigwedge_{sel} \mathcal{P}[sel]\right) \Leftrightarrow \mathcal{P}$

where in 1.ii/ sel is varying over the set of selectors of $\mathcal{P}$, and $\mathcal{P}[sel]$ is a subtype of $\mathcal{P}$ selected by sel. Also 1.ii/ shows the lifting mechanizm in SL: from the lower-level statements we can prove the next higher level statements and vice versa.

If a is a transition, operating on data-objects,

specified by $\mathcal{P}$ before the execution and by Q after the execution, then

$$\mathcal{P}\ a\ Q$$

describes the effect of a on data-structures and if $\mathcal{P}_a$ expresses the state of data-structures after a, then

1. iii/ $$\left(\left(\bigwedge_{sel}\left(\mathcal{P}[sel]\ a\right)\right) \Rightarrow Q\right) \Leftrightarrow \mathcal{P}\ a\ Q$$

Two comments should be made to 1.iii/. The first, that a constains a predicate π_a, and a body b_a hence in 1.iii/

$$\left(\bigwedge_{sel}\left(\mathcal{P}[sel]\ a\right)\right) \Leftrightarrow \pi_a \wedge \left(\bigwedge_{sel} \mathcal{P}[sel]\ b_a\right)$$

The second comment, that $\mathcal{P}[sel]\ a$ represents the effect of a on those objects, which are characterized by $\mathcal{P}[sel]$. So we have to distinguish three cases, denoted by:

1/ $\mathcal{P}[sel]\ a$

2/ $(\mathcal{P}\ a)[sel]$

3/ $(\mathcal{P}\ a\ Q)[sel]$

In 1/ a sub-structure of objects ot type $\mathcal{P}$ are expressed, in 2/ the sub-structure after the execution, hence in 2/ sel is varying on the selectors of Q, and, finally in 3/ "sel" is varying on the set of the selectors of $\mathcal{P}$ or Q. If $S(\mathcal{P})$ represents the set of selectors of $\mathcal{P}$, then

1. iii/ $$\left(\left(\bigwedge_{sel \in S(\mathcal{P})} \mathcal{P}[sel]\ a\right) \Rightarrow Q\right) \Leftrightarrow \mathcal{P}\ a\ Q$$

and for the cases 2/ and 3/:

1. iv/ $$\left(\bigwedge_{sel \in S(Q)} \left((\mathcal{P}\ a)[sel] \Rightarrow Q[sel]\right)\right) \Leftrightarrow \mathcal{P}\ a\ Q$$

1. v/ $$\left(\bigwedge_{sel \in S(P) \cup S(Q)} (\mathcal{P}\, a\, Q)[sel] \right) \Longleftrightarrow \mathcal{P}\, a\, Q$$

where the later is an analogous form of 1.ii/.

3.2.2. Verification of abstract control structures

In this section a_1, a_2 ... represents compound or simple Petri nets, "op" denotes the operations as in Chapter 1. We shall use deduction in the form

$$(\mathcal{P}_1\, a_1\, Q_1) \wedge (\mathcal{P}_2\, a_2\, Q_2) \Rightarrow \mathcal{P}_3\, (a_1\, op\, a_2)\, Q_3$$

Every ACS is defined by a wff, hence we have to verify only the formulas.

2. i/ If op = $\cup$ (union), then

$$\mathcal{P}_3 = \mathcal{P}_1 \vee \mathcal{P}_2$$

and

$$Q_1 \vee Q_2 \Rightarrow Q_3$$

2. ii/ If op = el-merge

then $\mathcal{P}_3 = \mathcal{P}_1 \wedge \mathcal{P}_2$ and $Q_3 = Q_1 \wedge Q_2$

and if op = merge then by the identity

$$N_1 \,\underline{\text{merge}}\, N_2 = \bigcup_{i,j}^{n_1,n_2} N_1^i \,\underline{\text{el-merge}}\, N_2^j$$

we have $$\mathcal{P}_3 = \left(\bigvee_i \mathcal{P}_{N_1^i} \right) \wedge \left(\bigvee_j \mathcal{P}_{N_2^j} \right)$$

and

$$Q_3 = \left(\bigvee_i Q_{N_1^i} \right) \wedge \left(\bigvee_j Q_{N_2^j} \right)$$

2. iii/ If op = exclusion ⟦

$$\mathcal{P}_3 = (\mathcal{P}_1 \wedge \neg \mathcal{P}_2) \vee (\mathcal{P}_2 \wedge \neg \mathcal{P}_1)$$

and

$$(Q_1 \rightarrow Q_3) \vee (Q_2 \Rightarrow Q_3)$$

Here that fact is expressed, too, that if $\mathcal{P}_1 \wedge \mathcal{P}_2$ = <u>false</u>, then

$$\mathcal{P}_3 = \mathcal{P}_1 \vee \mathcal{P}_2$$

2. iv/ For iteration $*$

$$((\mathcal{P}_1 \; a \; Q_1) \wedge (Q_1 \Rightarrow \mathcal{P}_1)) \Longrightarrow \mathcal{P}_2 \; (*a) \; Q_2$$

where $\mathcal{P}_2 \Rightarrow \mathcal{P}_1$ and $\mathcal{P}_1 \Rightarrow Q_2$

2. v/ For marking $*$ and for closure <u>do</u> ... <u>od</u> we have

$$(\mathcal{P}_1 \; a \; Q_1) \Rightarrow \mathcal{P}_1 \; (n \rightarrow a) \; Q_1$$

and

$$(\mathcal{P}_1 \; a \; Q_1) \Rightarrow \mathcal{P}_1 \; \underline{do} \; a \; \underline{od} \; Q_1$$

Also we have to use deductions for (atomic) assignements $x := e$

2.vi/ $$\mathcal{P}(x/e) \; x := e \; \mathcal{P}$$

where $\mathcal{P}(x/e)$ represents $\mathcal{P}$, but every occurence of x is substituted by expression e.

4. <u>Summary and open problems</u>

Our goal was to give a short description of language--phylosophy. This paper should be treated as a draft only: an illustration of the main concepts. Detailed description will be available elsewhere, including monitoring, etc.

The main results here

1. transformation of Petri nets into a programming language

2. allowing the contol as dynamic parameter
3. providing tools to verify programs in this complex situation.

In order to reach the real power of this language, we have to solve the following problems during the extension of these concepts:

4. In this form in a module-declaration either the data - or the control - part may be empty in the formal parameter list. Allowing both control and data structures as "returned values" would be a very useful extension. In the case of data it is a trivial fact, but for control structures it is not so easy. With this the concept of "computed control" or the "control of the control" would be solved.

5. In the language we have to formalize statements on abstract control structures. If a statement K stands for an ACS X, and L is for another one Y, what will be true for X op Y, where op is defined in Section 1? Probably we have to restrict somehow the language in this respect, because X op Y may share some recources, may have common control structures, etc., hence it is not well-structured.

6. Also we have to formalize statements on modules in the declaration, and during specification (call) too, and providing logical calculi, in which statements can be proved (like in Alphard).

With these extensions a really dynamic language would be described mostly for systems-programming purposes. One possible application is the dynamic microprogramming, we "manualy" checked these concepts, where the available resources registers, processors, etc. are varying, and a microprogram should have local control which is invariant against the hardware changes described as input parameters.

Dávid Gábor
MTA Számitástechnikai és Automatizálási Kutató Intézete
1111 Budapest, Kende u. 13-17.

ABSTRACT

ON THE BASIC CONCEPTS OF A MODULE LANGUAGE

Gábor Dávid

Our goal was to give a short description of language-phylosophy. This paper should be treated as a draft only: an illustration of the main concepts. Detailed description will be available elsewhere, including monitoring, etc.

The main results here

1. transformation of Petri nets into a programming language
2. allowing the control as dynamic parameter
3. providing tools to verify programs in this complex situation.

REFERENCES

1 Dávid, G., Keresztély, S., Sárközy, A.: Microprogram Synthesis by Theorem Proving, II. Hung. Comp. Sci. Conf., 1977. part 1, pp 291-310.

2 Dávid, G.: Structured Automatized Design of Microprograms, EUROMICRO 1978, in preparation.

3 Dávid, G., Keresztély, S., Losonczi, I., Sárközy, A.: Logic-based description of the architecture, Proceedings of Winter School on Microcomputers, 1978 MTA SzTAKI Report in print

4 - : Automatic Microprogram Synthesis, as in 3

5 Kotov, V.E.: Concurrent programming with control types, IFIP TC.2, Working Conference on "Constructing Quality Software", 1977, North Holland

ALARM ANALYSIS: AS A FORM OF SECONDARY DATA PROCESSING

Laura Bürger

1. INTRODUCTION

A form of secondary data processing will be discussed in the paper. The data to be processed are gained from the computerised data acquisition and control system of the WWR-SM research reactor in the Central Research Institute for Physics, /Budapest, Hungary/ and are used to construct an advisory system supporting the work of the operators [7]. The applied computer is an R-10, with 64K byte core and 800 Kbyte disc memory. The real-time peripheral equipment contains 128 analogue and 12x16 bits digital measuring channels. The applied software system is the PROCESS-24K, described in [5].

2. SECONDARY DATA PROCESSING

Secondary data Processing - in our definition - is a rearrangement of the primary, still disarrayed data base into an ordered form, according to some organizing principles. It can be considered as a transformation, where many kind of transfer functions may be applicable depending on the desired goal. This transfer function is surely different in the paper industry, in a chemical process or in a nuclear reactor. In the later case different variations may emerge such as trend analysis, state estimations, etc. Our transfer function serves the goal of the alarm analysis.

3. ALARM ANALYSIS

The idea of the analysis of the alarm states has arised at computerised process control systems planted beside large, valuable and sophisticated equipments /missile control, chemical plants, nuclear power plants/.[1, 2] In these places the volume of the measured data may be more than a thousand in every second. If a disturbance occurs many various alarm signals, arriving from different parts of the workshop, are announced to the operator, - this is the so called "alarm state". The operator has to recognize the situation, to decide and to intervene during a very short time, moreover to find the proper action in a complicated situation - when he is in a stress condition - is very difficult. To support the operator's decision by an efficient advisory system is highly desirable. Such a system collects and analyses the alarm signals and gives some information on the possible causes and about the expectable consequences. The decision on the necessary intervention is the operator's duty, at this point. However, - supposing that a closed loop process control system exists - the alarm analysis may intervene automatically. The output of the analysis may be used to the modification of the control algorithms in some, well defined, emergency states.

The analysis of the alarm states may be executed in two directions: toward the causes and toward the consequences. The international literature is not unified in the question: which direction has the greater importance? The analysis of both directions is realized only in the most developed systems. [3] We have just accomplished the analysis towards the causes.

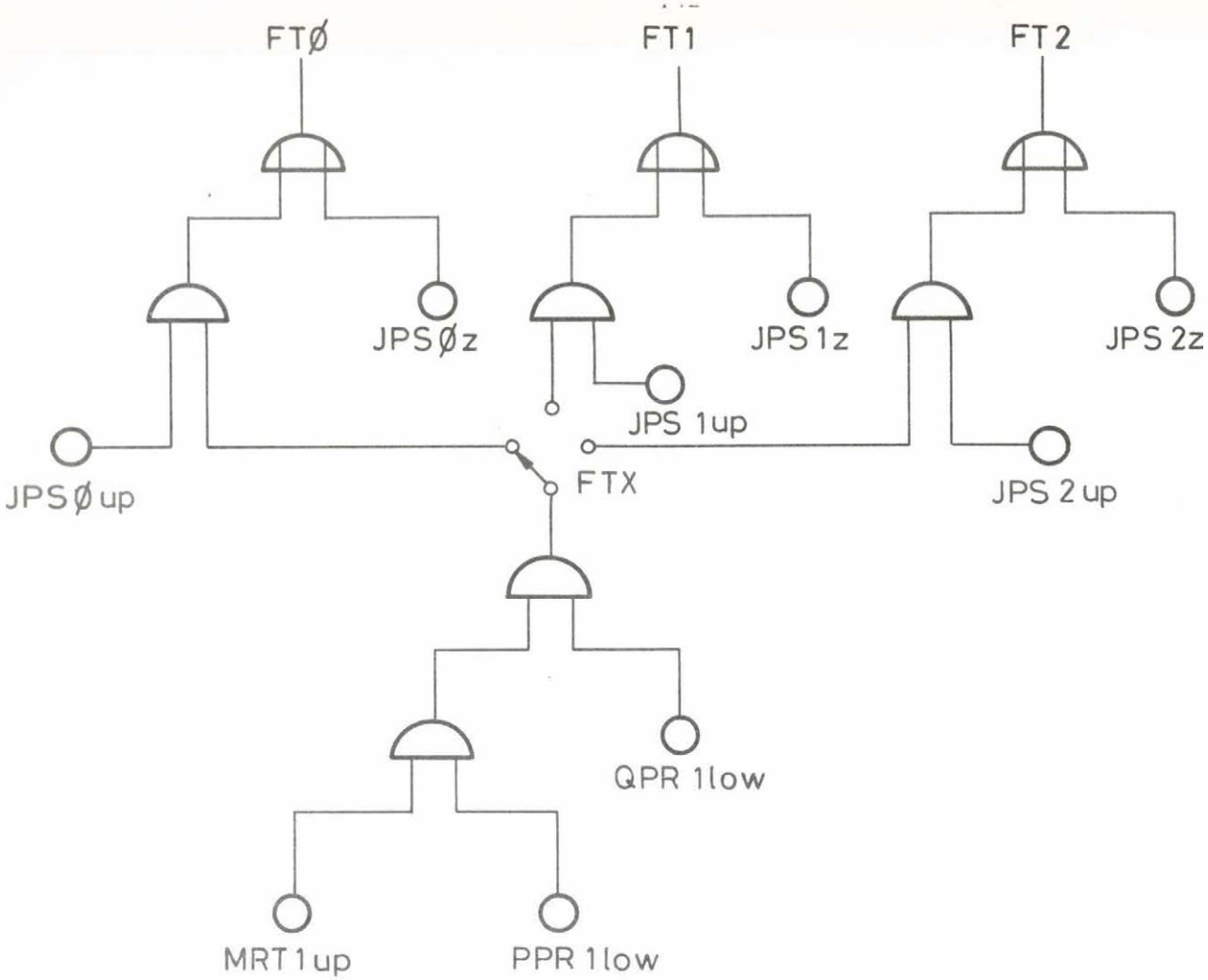
3.1. PREPARATION OF THE ANALYSIS

The first task is to discover the logical connections among the various alarms in the case of a well defined failure. It means, that the transfer function has to be defined for the secondary data processing. Each failure has its own transfer function, so it is possible to separate only those failures whose transfer functions are different from the others. The variables of these functions are alarm and state signals and their time sequences. The relations among the variables are described by logic tree structures. The primary failure to be identified is on the top of the "alarm tree", the alarm signals characterizing the failure are on the terminal nodes on the lowest level, and the logical connections among them are represented by the intermediate nodes. /Fig. 1./ /The task is similar to the fault tree analysis known from the reliability calculations of large systems./

To defind and to collect the alarm trees of a large, often sophisticated system /as in our case/ is a very complex task. A very detailed "failure mode and effect analysis" is necessary to look for all the expectable alarm situations. The construction of the alarm trees from the revealed connections then can be formalized. Some programs on higher level languages are known to perform this task. [4]

3.2. ON-LINE ALARM ANALYSIS IN PROCESS-24K

The alarm trees gained from the system analysis of our WWR-SM type nuclear research reactor are collected into the "Alarm Library" of the PROCESS-24K, and they are stored in a binary tree structure. The logical connections represented by the intermediate nodes of the trees may be AND, OR, and NOT Boolean functions. In the course of the analysis the nodes of a tree have to be examined systematically, so



MRT1 – Water level in the reactor tank
PPR1 – Pressure in the primary coolant cct.
QPR1 – Primary water flow
JPSi – Current of the i pump
JPSiz – Current of the i pump = Ø
FTX – Short circuit at the pumps
FTi – Short circuit at the i pump

Fig. 1.
An example for alarm trees

that each node is traversed only once. The analysis starts at the terminal nodes and process to the higher levels.

The place of the alarm analysis in the process control software system of the WWR-SM - PROCESS-24K [5] - is shown in Fig. 2. The information about the process variables is collected in the "Primary Data Base". The "Primary Data Processing" performes the habitual processing tasks: conversion into technical units, linearization, filtering, limit checking etc. If a limit is violated, the alarm analysis will be initiated. The tasks of the alarm analysis are distributed among three programs /see Fig. 2./

- ANAL is the name of the core resident task, which performs the analysis /priority level 15/,
- ALDYS carries out the presentation of the alarms /priority level 11/,
- ALGEN serves the generation and modification of the alarm library /running in the background of PROCESS-24K/.

Only the operation of the ANAL program will be discussed in this paper in rather detailed form.

If an alarm event has occured, the ANAL program is initiated by the "Primary Data Processing". Alarm event occurs if the value of an analogue variable has passed its trip level, /or it has returned within its limits/ or if a bit of a digital input has changed its state. As the first task, the ANAL program updates the so called "Secondary Data Base", which forms the basis of the analysis. The next task is to analyse the "alarm situation", the existing "alarm pattern" using the second part of the library: the Tree Descriptions. The program tries to find the alarm tree adequate to the alarm pattern. The appropriate tree is the tree, on which it can reach the top, last deduction, the primary cause.

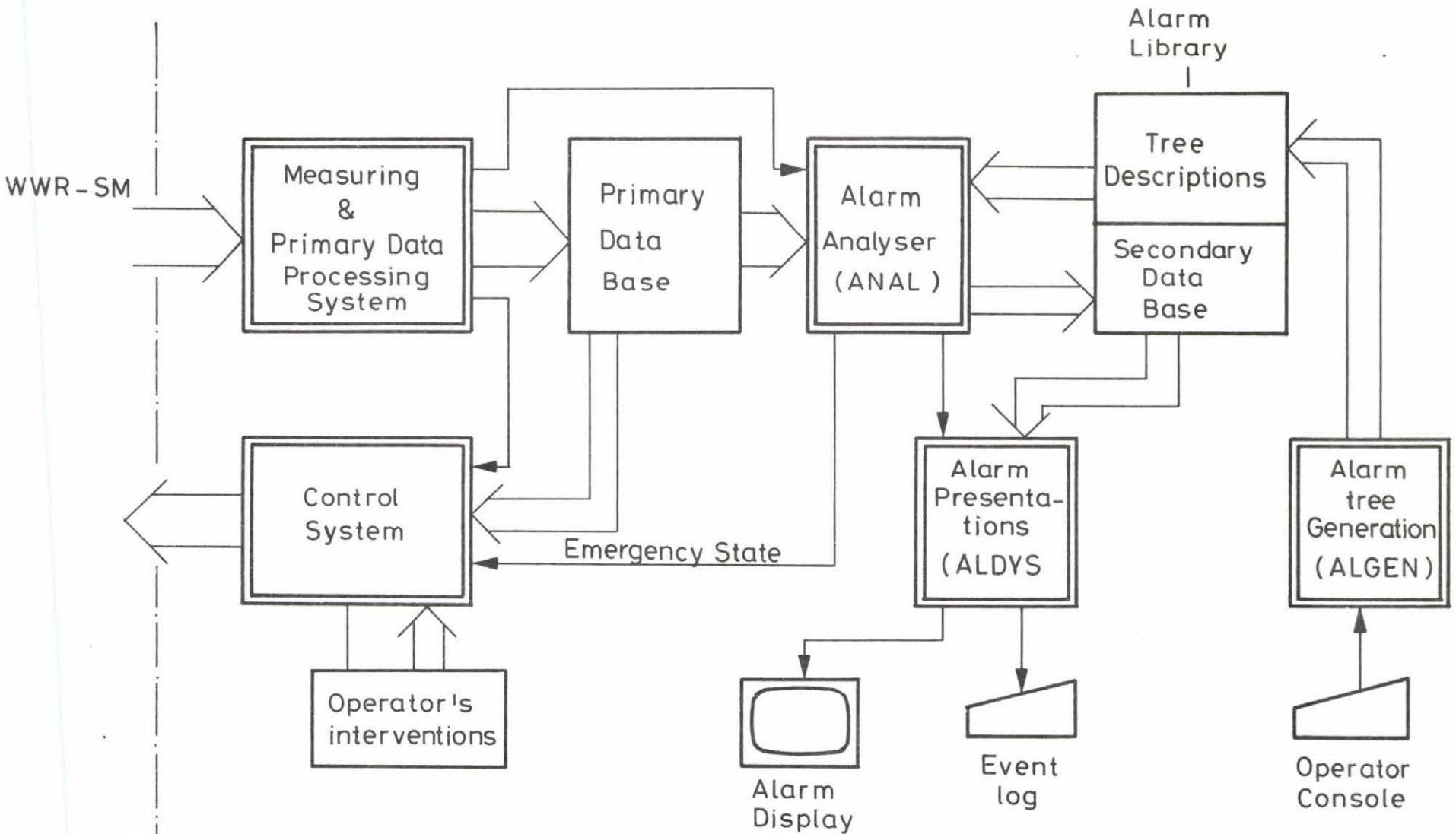


Fig. 2.
Place of the alarm analysis in PROCESS 24-k

This is the adequete transfer function for updating the Secondary Data Base. For the sake of unity, the last /and intermediate/ deductions - which are the results of the Boolean operations specified in the nodes - are considered as alarm too, and are stored in the Secondary Data Base as well. These are the so called "deduced alarms".

The last duty of ANAL program is to initiate either the alarm presentation /ALDYS/ program, or the Control System - or both - according to the specification of the last node.

3.3. THE ALARM LIBRARY

The alarm library has two main parts, as it was already mentioned previously:

- library of the alarms: the Secondary Data Base,
- the tree descriptions.

The Secondary Data Base contains the result of the secondary data processing. The tree descriptions are the transfer functions, which forms the basis of the transformation of the primary data. These transfer functions are Boolean functions, consequently the secondary data base is a set of logical data.

3.3.1. Library of the Alarms

The secondary data may origin from three type of alarms:

- alarms of analogue variables: "analogue alarms",
- bit changes in digital inputs, i.e. changes of state: "digital alarms",
- the results of the analysis of the alarm trees: "deduced alarms".

The logical value of an analogue alarm will be equal to one, if the analogue variable concerned is in the alarm state. The logical values of the digital and the deduced alarms are obvious.

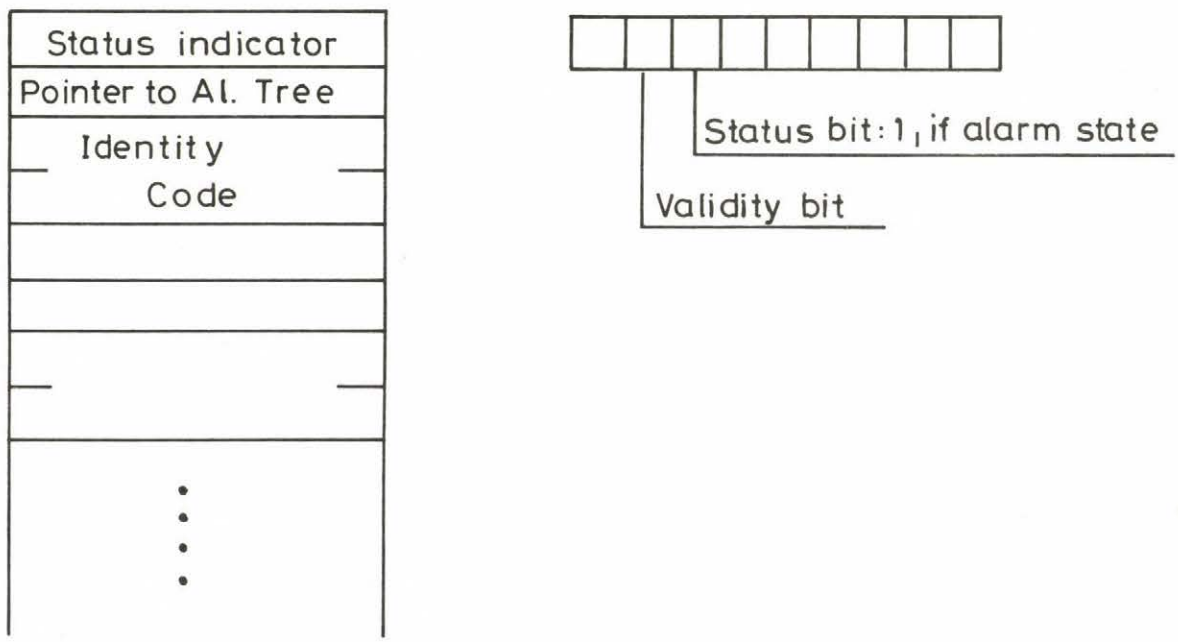


Fig. 3.

Structure of Secondary Data Base.

The data to be stored about an alarm is collected into a four byte long description /Fig. 3./. The logical value is represented by the "status bit" in the first byte. If the value of the validity bit - which may be set from the Primary Data Processing - is "0", the alarm does not take part in the analysis.

The second byte contains a pointer to the alarm tree, on which this alarm is a node. The last two bytes are necessary to the identification of the alarm in the system.

To speed up the analysis, the Secondary Data Base is core resident. The size of this part of the alarm library is 2,5 Kbyte, which is room enough for 640 different alarms.

3.3.2. Library of the Alarm Trees.

This second part of the alarm library contains the tree descriptions /Fig. 4./. Each description can be divided into two main parts: the Tree Description Head /TDH/ and the administration of the nodes.

The Tree Description Head contains a pointer to the first node of the traversal and - if it exists - the identity code of the emergency control algorithm, which have to be started if the last deduction /the deduced alarm on the top/ becomes true, and some working cells necessary in the course of the analysis.

The scanning pointers defined to the individual nodes indicate the order of traversal. The so called End Order Traversal /EOT/ is used [6]. The applied simple rule is the following:

- traverse the left subtree,
- traverse the right subtree,
- traverse the root.

The identity code in the node description identifies the alarm belonging to it.

An alarm may appear on several trees. In the four byte long description of the Secondary Data Base there is room only for the pointer to the first tree. The pointer to the

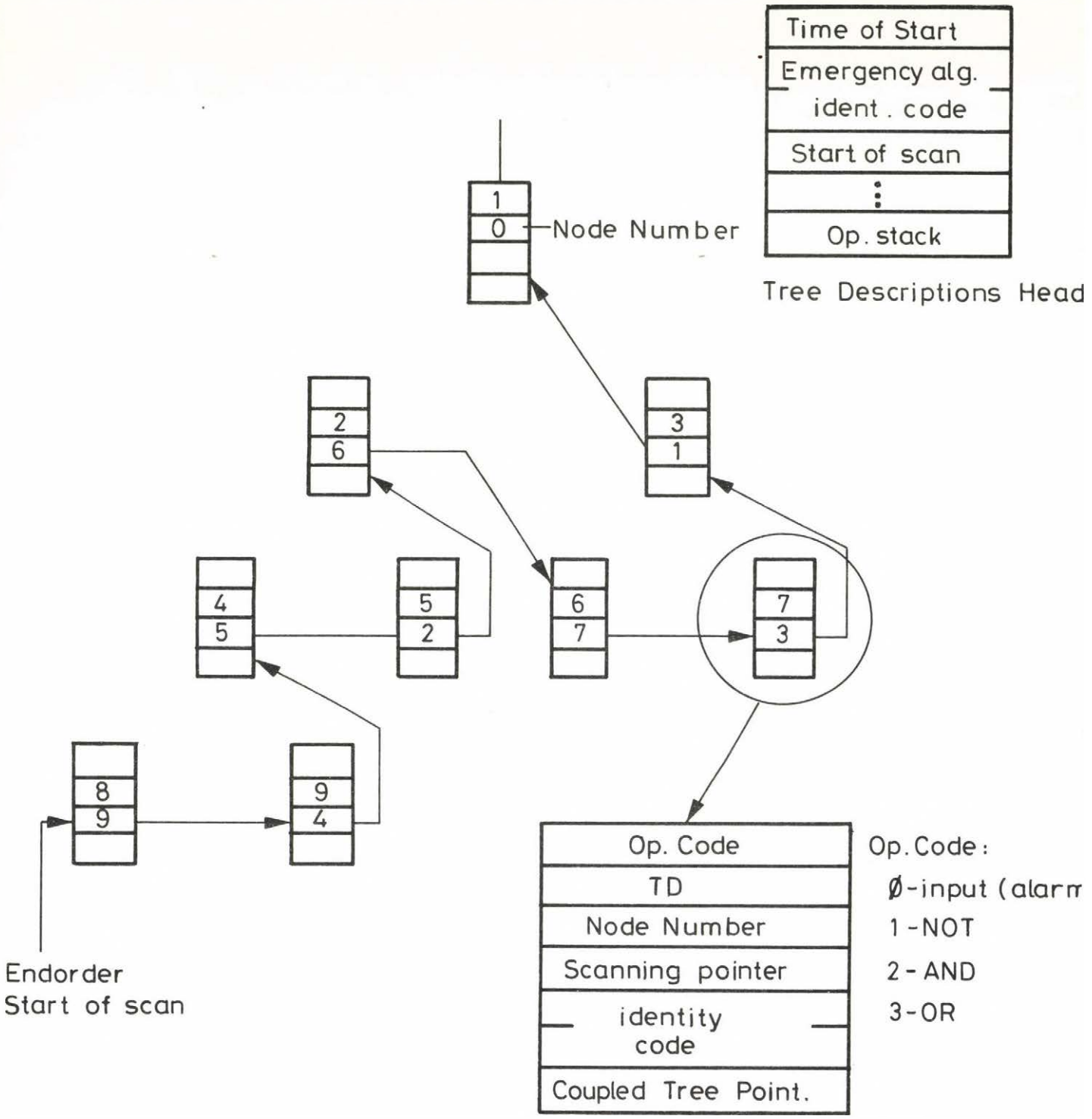


Fig. 4.
Description of an alarm tree

next /second, third, .../ tree is included in the node administration of this tree. This is the "Coupled Tree Pointer".

A very essential data is included in the cell TD /Time Delay/. It contains the information about the expectable chronological order of the appearance of the alarms. The alarms assigned to a given tree may occur in different time due to

- the different propagation of the effect of the disturbances, and
- the different cycle times of the measuring groups.

Considering this fact for each alarm tree one has to specify the probable first occuring alarm and a Time Delay /TD/ to every subsequent node. The advantage of this procedure is twofold:

1/ In the case of a rather large alarm tree such a situation may occur, when alarms exist already on several nodes of the tree, but additional alarms which may come /or not/ later, are still necessary to the last deduction. The analysis has then to be suspended for the TD of the expected alarms. If some useful deduction has been found from the analysed part, it can be presented to the operator already in this intermediate stage.

2/ The definition of TD offers the possibility of time sequential analysis of the trees; - i.e. distinction can be made among the alarm trees, whose alarms are different only in the sequence of their occurence.

The tree description part of the library is disc resident. Only the description of the just analysed tree is in the core. The size of the disc area reserved for the trees has to be defined in the course of the system generation.

3.3.3. Generation of the Alarm Library

The generation of the alarm library is supported by the ALGEN program. The generation of the trees, definition the scanning order, the time delays and the specification on their presentation can be made by the use of this system program. The store assignment of the Secondary Data Base is elaborated by ALGEN as well. Its use does not need any programming knowledge.

ALGEN is running in the background of the PROCESS-24K, so the generation and modification of the alarm library is possible during the operation of the whole process control system.

4. ALGORITHM OF THE ANALYSIS: ANAL

The flow-chart of ANAL can be seen in Fig. 5.

The ANAL program can be initiated by two tasks:

a/ by the Primary Data Processing at the occurance of a new alarm,
b/ by the timer in every second.

Tasks of the ANAL initiated by a new alarm are:

a/ to separate the alarm events participating in the analysis. The remaining alarms are passed to the ALDYS program for presentation;
b/ to evaluate the new logical value of the alarms and to update the Secondary Data Base;
c/ to put the pointer of the tree containing the alarm into the ANAL Waiting List /AWL/.

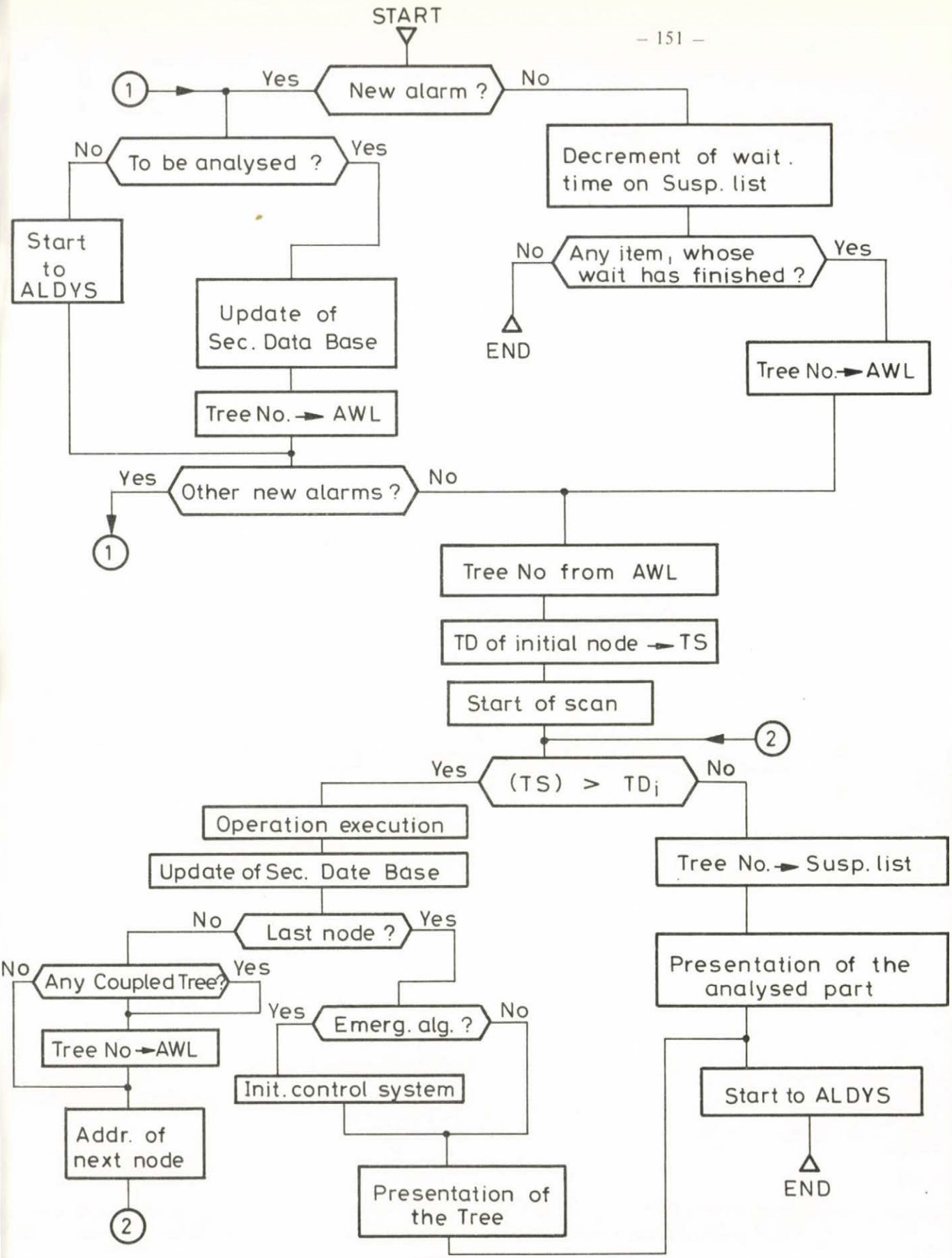


Fig. 5.
Flow chart of ANAL

Tasks of the ANAL initiated by the timer are:

a/ to decrement the waiting time on every item of the list of the trees, whose analysis have been suspended /Suspension List/;

b/ to put the pointer of the tree/s/, whose waiting time has become zero, into the AWL.

Having been finished these tasks the tree analysis is started. The main steps of the analysis are:

a/ to look for the node corresponding to the alarm event initiating the analysis. The Time Delay defined for this node will be deposited into the Time of Start cell of the Tree Description Head /TDH/;

b/ to start the analysis from the node inidicated by the Start of Scan in TDH;

c/ to analyse the consecutive nodes in succession indicated by the Scanning Pointers. Before the execution of the operation prescribed to the node the program checks the value of the node TD;

d/ if it is larger than the Time of Start, the analysis will be interrupted and the tree will be written on the Suspension List;

e/ if the TD is smaller than the Time of Start, the logical operation defined by the Op. Code of the node will be performed in the operational stack of the TDH, and the result will be written back into the Secondary Data Base;

f/ if there is a coupled tree defined to the node its pointer will be written in the AWL;

g/ having reached the last node,the top - the emergency control algorithm - if it exists - is initiated and the description of the tree - in a suitable form for presentation purposes - is sent to the ALDYS program.

5. PRESENTATION OF THE ALARMS

The task of alarm presentation can be devided into three cathegories:

- messages to be written only into the "event log",
- messages concerning those alarms, which have to be written into the screen of the "alarm display" and into the "event log",
- alarm pictures /trees/ to be presented on the alarm display.

Only the data of the last two cathegories have to be stored. There are two lists on the disc: the "alarm list" and the "picture list" where the identity codes of the messages and pictures are stored. The operator can select a list for presentation on the CRT by push buttons of his control desk. Another push buttons are provided to turn the "pages" back and forth in the content of the lists.

REFERENCES

[1] D. Patterson: Application of a computerised Alarm-Analysis System to a Nuclear Power Station. Proc.of IEE, Vol. 115, No. 12. 1968.

[2] D. Welbourne: Alarm Analysis and Display at Wylfa Nuclear Power Station. Proc. of IEE, Vol. 115, No. 11. 1968.

[3] R. Grumbach, H. Hoermann: A Program System for Plant Disturbance Analysis. Report of Techn.University, München, 1976.

[4] E. Holló, J.R. Taylor: Algorythms and Programs for Consequence Diagram and Fault Tree Construction. Report Riső-M-1907, Dec. 1976.

[5] L. Bürger, Z. Csörnyei et. al.: PROCESS-24K - An Efficient Process Control System. Report KFKI-1978-17

[6] D.E. Knuth: The Art of Computer Programming. Addision-Wesley Pub. Co. Vol. 1.

[7] J. Péter, E. Végh: Data Acquisition Program for a Nuclear Research Reactor. 2nd Int. Conference on Centralised Control Systems. IEE publ. No. 161. pp. 131-134, London, March 1978.

Laura Bürger
Central Research Institute
for Physics
H-1525 Budapest, P.O.B.49.

THE STRUCTURE AND THE EFFICIENCY OF THE PROCESS-24K PROCESS CONTROL SYSTEM

E. Végh

1. INTRODUCTION

In the early 70s the utilization of digital computers for industrial process control increased dramatically and this in turn initiated a great development in industrial real-time programming. Although in the early stages the assembly coding predominated the industrial application area, now various types of real-time languages are used at almost every installation. The main reasons for this lie in the growing size and complexity of the problems and in the drastic decrease in the price of the computing hardware.

Two tendencies in the use of high level real-time languages in the industrial environment can be observed. One of these trends is in favour of existing widely accepted high level languages, and provides real-time extensions to them. The other prefers new problem oriented languages which allow the user of an industrial computer to develop his system easily without having a detailed knowledge of the used computing means.

Universal high level languages, which are completed with real-time extensions, are FORTRAN and BASIC.

A considerable number of problem oriented languages have been developed recently, e.g.: INDAC, PROCOL, LTR, PEARL, CORAL, etc.

With such languages the program writing time and errors are reduced since the real-time problems are solved by the language. The debugging and the program modification are quite simple and well decumented. The language efficiency is considerably good, about 1,3-1,5 compared to assembly programming. The experience of real-time programming in an industrial environment has proved that a problem is better solved by a process man, who knows the process well but has a very limit knowledge of programming, rather than by an experienced programmer, who may not understand the process. This fact has initiated the development of simple but effective operating systems, which are not general purpose systems but rather process oriented ones. In general a process oriented operating system incorporates the compiler of a given high level language and tries to simplify the programming in every possible way.

PROCESS-24K is a stand-alone process oriented operating system for the R-10 computer [1]. It is based on the PROCESS-8K system [2] but its efficiency is much more higher, moreover it contains a further abstraction level which does not exist in the predecessor system. However PROCESS-24K maintains an upward compatibility with the predecessor system, i.e. each user program of the PROCESS-8K can run in our system without any modification but it is not true in the other sense.

2. SYSTEM ARCHITECTURE

In the PROCESS-24K system 4 different layers can be distinguished, each layer has a specific structure and interface. These layers are built on each-other hierarchically and in general only the neighbouring layers communicate with each-other. Every layer has a special dependency from the actual process and it is weaker in the lower ones than in the uppers. These layers are the following /see Fig. 1./:

- operating layer, which contains all of the programs needed to the operation of the computer hardware /peripheral handlers, swapping control, buffer system, recovery procedures etc./
- data acquisition and control layer, which consists of timing, measurement organization, primary data processing, data base organization and data presentation
- data analysis layer, which means trend and alarm analysis, alarm presentation
- adaptive control layer, which can reconfigurate the measurement / control tasks.

From an information processing point of view this system provides two images of the outer world /i.e. the controlled process/. The data acquisition layer up-dates cyclically a data base which is a more or less unstructed picture of the process, containing every measured item of information without any deduction /except for validity checking/. The data analysis layer generates a structured picture of the process so this image depends not only on the measured quantities but on the ordering principle too. Consequently this picture is more abstract and condensed than the former one; at this level the process is described by state matrices.

The operating layer consists of every software means to operate the computer hardware, that is

- the handlers of the different peripherals
- the used buffer system, overlay and swapping technique, background organization,
- error recovery procedures, activation and initiation of peripherals, different types of checking.

This layer does not depend on the actual process, but only on the computer hardware. In Fig. 2. the hardware configuration, which can be controlled by the operating layer can be seen.

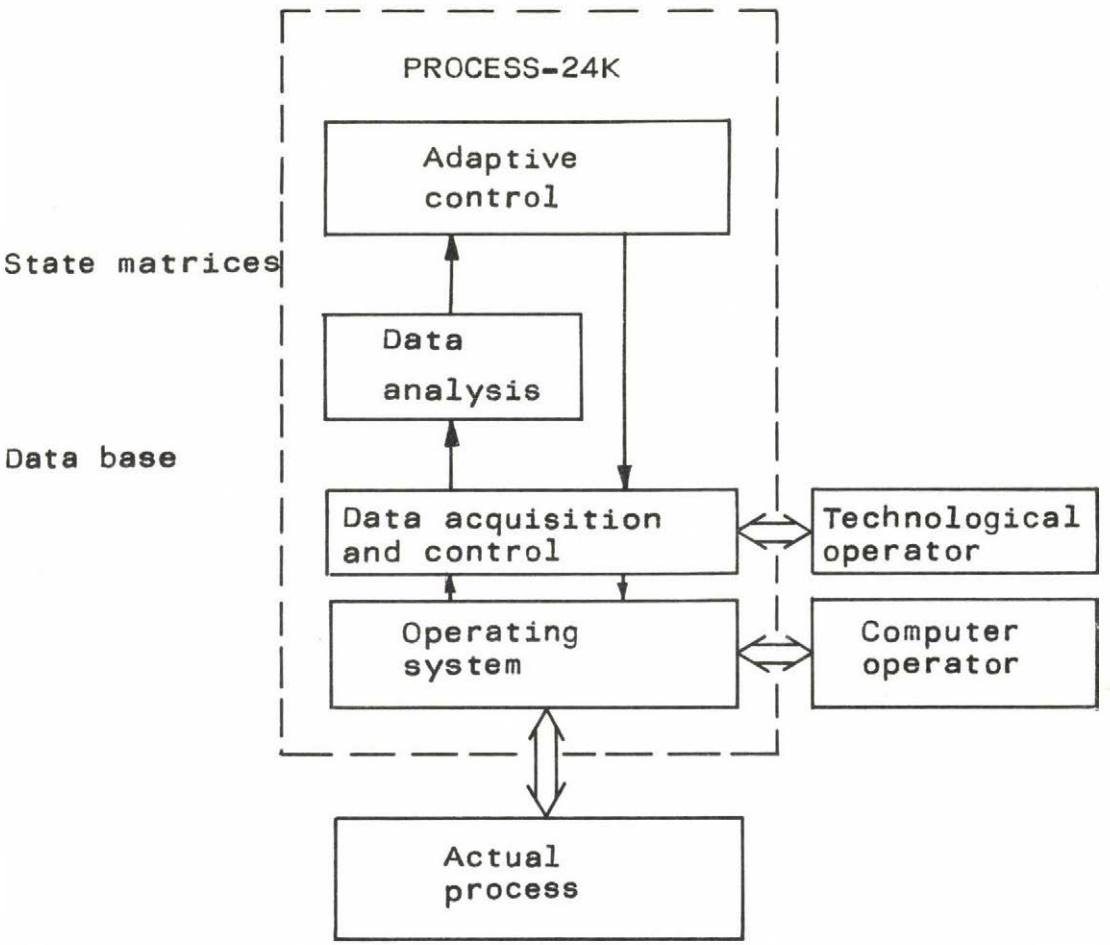


Fig. 1.

Structure of PROCESS-24K.

The kernel of this system is an R-10 central processor with 48 kbyte core memory and floating point arithmetic unit. Because the complet Process system is much more larger than the core memory, a fixed head disc with 800 kbyte capacity is also a fundamental part of the system.

The computer operator has an access to the operating layer. He can replace a peripheral unit by an other one and he can use of the conventional peripherals for program development. There are about 15 utility at this disposal.

All of these programs serve the checking and the development of the running real-time control system. In this sense the operating layer is a dedicated one, the programmer can develop or modify the system and he has all the aides to do it, but he can not use the background as a general purpose computing facility, that is he can not translate or execute for example a FORTRAN program.

The data acquisition layer creates data base, that is an inner picture of the measured environment and informs the operator on the state of this data base. This problem incorporates the following tasks:

- to measure the environment
- to process the measured data, that is scaling, filtering, checking etc.
- to display certain measured variables on lamps, on numerical indicators, on analogue registrators etc.
- to provide an interface for the technological operators
- to generate logs of different types
- and to execute the described control actions.

All of this problems are solved by table controlled subsystems and the user has only specify the operation of each subsystem by filling out its control table.

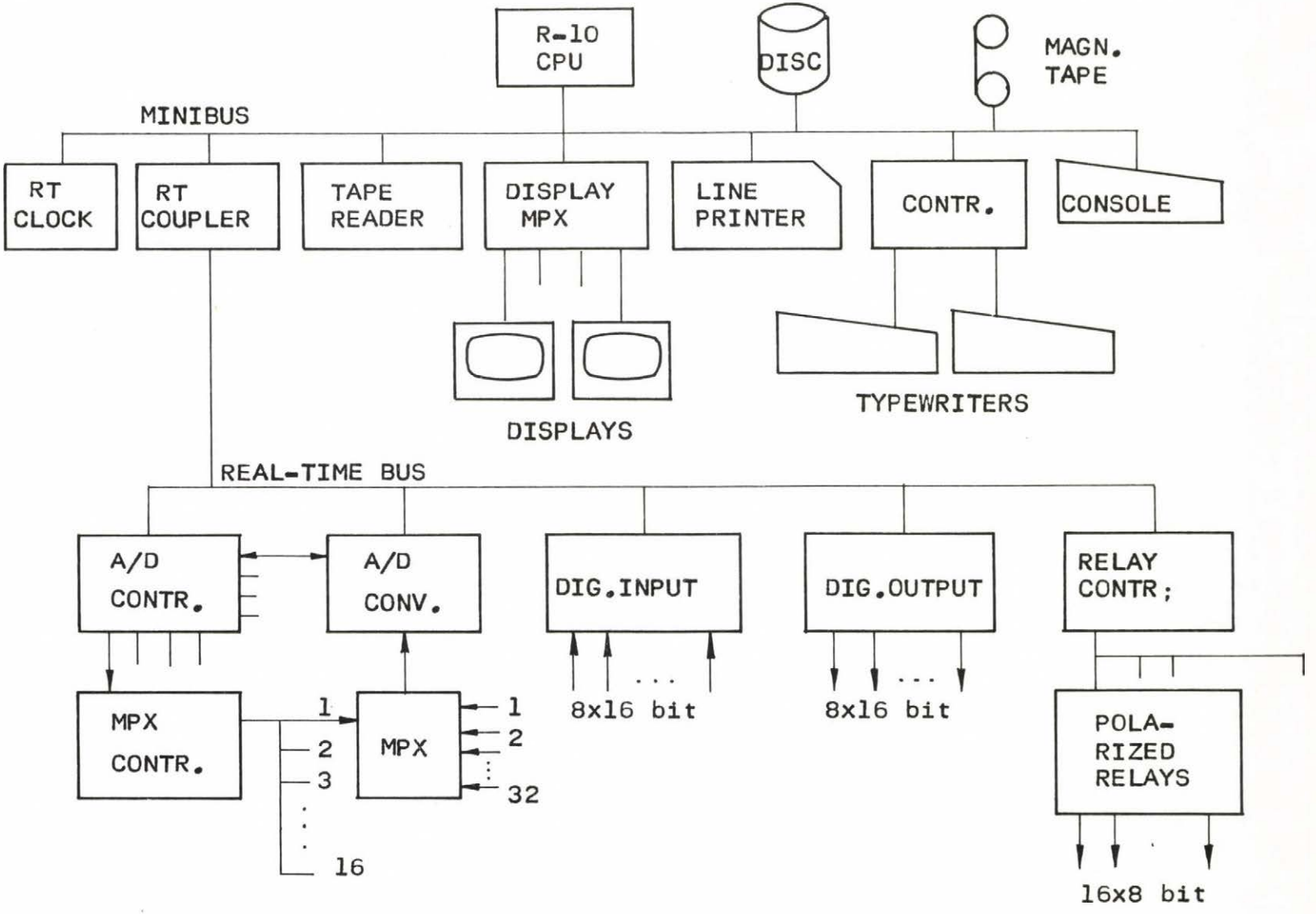


Fig. 2.

Hardware configuration for PROCESS-24K.

In the core resident data base every variable has a value and a flag. The value is a 4 byte long floating point number in the case of real numbers /that is analogue variables/ or a 2 byte long hexadecimal number in the case of integer numbers /that is digital variables/. In the both case the flag is a byte, every bit of which reflects something about the given variable, for example

- the measureement is active or not,
- the control action is permitted or inhibited,
- the measured quantity is valid or not, etc.

The core resident data base has an image on the disc. This image contains the initial values of the variables which is essential during control. This image is loaded into the core during the Initial Program Loading phase, that is when the Process system is started.

The primary processing is described by the PROCESS language. In this language there are 61 different executable instructions. They are

- arithmetic: addition, subtraction, multiplication, division and power function
- logical: and, or, exclusive or, rotation etc.
- data transformation, that is linear transformation, linearization, digital filtering, validity checking etc.
- branching, unconditional jump, conditional jumps, where the condition may be
  - the validity
  - the result of the alarm limit checking
  - the logical value of a given bit of the accumulator etc.
- internal functions, for example: exponential, logarithmic, square roof functions, absolute value, integer value, etc.

This function repertoire is not the same as it is usual in other high level languages. For example there are no sine, cosine, tangent functions, because they are rarely needed in process control applications, but there are unusual functions for example floating functions which transforms an integer value into a real number, for it is need when one use a digital/ analog converter.

- control instructions realize the generally used control actions, that is an analogue or a digital, or a time modulated digital output can be described. The PID instruction realizes a PID control action.
- finally there are program transfer instructions, subroutine call and return and cycle organizing instructions.

It is not enough to create a data base, but in a control room fast data presentation is also needed. This data presentation is carried out by

- mimic display tables,
- by numerical indicators,
- or by lamps built into a control desk.

For this purpose PROCESS-24K can display every second maximum 32 variables automatically. The variables to be displayed can be either real or integer number. If it is integer, the present value of the variable is sent to the specified digital output, if it is real, first its value is converted into a 4 digit BCD code and this code is sent to the output. Such a way numerical indicators /for example Nixie tubes, 7-segment displays/ can be drived directly.

In addition to the mentioned tasks, the primary data processing generates logs of different type such as

- periodical plant log,
- event log,
- post-mortem log.

The technological operators can maintain a communication with this level through alphanumeric display units. There are 20 different operator commands, by which the technological operator

- can interrogate the last measured value of a variable,
- can activate/inactivate a measuring or control action
- can alter alarm limits, validity limits, etc.

The alarm analysis layer is described in detail in the literature [3].

In the life of a plant, the goal of the control is not the same

- when the plant is started,
- when it operates at a normal working point,
- or during an emergency situation.

For this reason the adaptive control layer accomodate itself to the given situation. PROCESS-24K can provide

- a single measuring and control action, when a given event occur in the plant, or
- can reconfigurate all the real-time tasks.

Single actions are initiated by external interruptions. 8 different interrupts can be defined and each can has different groups to measure. On the other hand, when an emergency situations occurs, the structure of the cyclic tasks has to be changed. It is initiated by the alarm analysis layer and the adaptive control layer changes the control table of the timer in which the cyclic tasks are prescribed. Such a way the system will execute other groups with other repetition time, that is the whole real-time structure is changed.

3. THROUGHPUT OF THE SYSTEM

The performance of PROCESS-24K was analysed in a system with 70 analogue variables and with 11 measurements/sec information rate [4]. It was found that the updating of one analogue variable needs 5-6 ms of CPU time. This time includes

- control of multiplexors and A/D converters,
- converting the measured quantity into a floating point number,
- scaling,
- comparison against alarm limits,
- exponential filtering,
- storing in the data base,
- houskeeping of the data acquisition layer.

The real-time measuring hardware of the R-10 computer uses slow A/D converters of integrating type with a considerably good noise suppression /120 dB at 50 Hz/. The maximum data rate of this converter is 30 measurements/sec. PROCESS-24K can control 4 A/D converters at the same time, so a maximum of 120 measurements/sec can be acheaved. This maximum information rate needs 120x6 = 720 ms, or 72% CPU time.

The overhead of the system /i.e. timing and refreshing the digital outputs every second/ is 1,5-2%.

Consequently, in the case of the maximum information rate, about 25% of the CPU time is available for operator communication and data analysis which seems to be a reasonably good value. The main characteristics of PROCESS-24K are given in the following Table.

Max. number of variables	2304
Max. number of measurements	1920
Max. information rate /meas./sec/*	120
Max. number of self-holding digital outputs /bit/	2048
Max. number of refreshed outputs /bit/	512
Floating point representation with	
length of mantissa /bit/	24
length of exponent /bit/	7
sign bit	1
Time resolution /ms/	50
Max. number of post-mortem samples	256
Max. number of alarms	640

Main characteristics of PROCESS-24K.

---

* determined by the controlled A/D converters

REFERENCES

1 L. Bürger et.al.: PROCESS-24K - an efficient process control system. Report KFKI-1978-17.

2 B.K. Kovács: Ein programsystem zur Messwerterfassung und Prozess Steuerung für den Digitalrechner ES 1010. VIDEOTON Software Nachrichten. 1975/2.

3 L. Bürger: Alarm Analysis: As a Form of Secondary Data Processing. Presented in this proceedings.

4 J. Péter, E. Végh: Data acquisition program for a Nuclear research reactor. Conference on Centralised Control Systems. 1978. IEE Publication Number 161. pp. 131-134. London

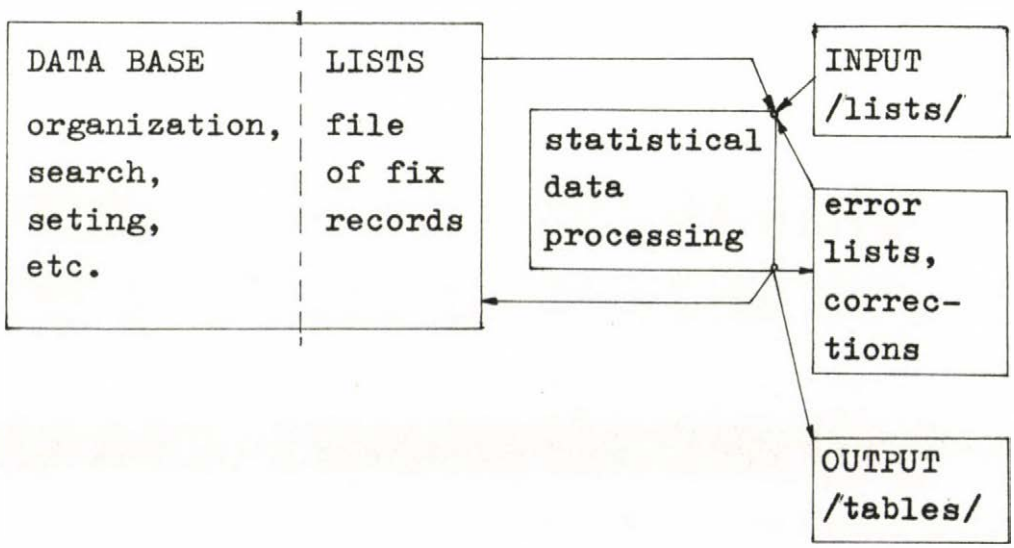
Endre Végh
Central Research Institute
for Physics
H-1525 Budapest, P.O.B. 49.

STATISTICAL INFORMATION SYSTEM WITH HEALTH SERVICE APPLICATION

M. Ruda

Statistical data processing problems have specific concepts and methods when used in large data base system. Investigating these concepts and questions means can be acquired which make the establishment of general statistical data processing systems possible.

Statistical data processing must often be carried out in relation to a given situation, independently from the generation process of the data base. A possible solution of the relations between data bases and statistical data processing is given on the next figure:



The information system built at the Probanility and Mathematical Statistics Department of the Computer and Automation Institute Hungarian Academy of Sciences was based on the following concepts:

1. Data control and creating supplementary data,
2. Sample selection and decomposition of the data system,
3. Establishing normal forms,
4. Creation of new data types /statistical from individual ones/,
5. Apparition of data /in tables/,
6. Administrational tasks.

The system /SIS77 - Statistical Information System 1977/ proved an optimal tool under general assignments regarding its successful application to the morbidity investigations in Hungarian hospitals.

What is guarantee the effectual function? We can enumerate the following considerations:

1. The completeness of the system /see the above points 1-6./,
2. The subsystems work in a general form /parameter control/,

3. We can use the subsystems independent of each other /modul structure/,

4. There are varied possibilities in the assortment, we can build in new subsystems,

5. We optimize the tasks which are appear most frequently /see eg. [6], [7]/, for the sake of that there is a generally applicable program optimizing method /see [9]/.

Given these major considerations, the subsystems assigned to the above mentioned notions 1-6. are /see [8]/:

1. The system provides a general data cheking and conversion facility /see [1o]/.

2. The partitioning of the data system yields a processing which requires less memory capacity and processing time. Selection of the data subsystems can be made by any arbitrary logical condition.

3. The involvement of normal forms enables an unified data system management /see eg. [1]/.

4. Results obtained by statistical data processing do not contain the data of individual items but those of typifying ones. Thus the system consisting of these individual data must be transformed into that of statistical data. In this

system the data appear alredy in forms of frequency characteristics, code values' totals, quadratic sums, product sums, etc., referring to each type. It is these data, statistical evaluations and mathematical statistical processing are based upon. So in statistical information systems it is not advisable to apply the languages developed particularly for handling and query processes of individual data items.

5. The output tables may contain any of the following data: frequency values, totals, different complementary values /such as percentages, ratios - e.g. average performance - subtotals, rates - e.g. characteristics per capita/, the basic data of statistical analysis /quadratic sums, product sums/, results of statistical evaluation /mean, deviation, correlation, etc./, graphic visualization.

6. The administration subsystem which supervises the whole system's operation has a file handling facility, and enables to any data grouping, contracting, deletion and storage /e.g. for population data/. Tasks for this, there are a few easy-to-handle input programs made.

During the system's development, our long experiences of data processing in the field of hospital applications /see [2], [3], [5]/ have been utilised, as well as the different mathematical statistical considerations /see [4], [6]/.

REFERENCES

[1] Codd E.G. , A Relational Model of Data Banks, CACM, 13., 197o.

[2] Csukás A., Greff L., Krámli A., Ruda M. , Statistical and data processing concepts in connection with the hospital morbidity survey, /in Hungarian/ 4. Neumann Conference, Szeged, 1973.

[3] Csukás A., Greff L., Krámli A., Ruda M. , An approach to the hospital morbidity data system development in Hungary, Symposium on Medical Data Processing, Toulouse, 1975.

[4] Garádi J., Krámli A., Ratkó I., Ruda M. , Application of statistical and computing methods for hospital morbidity studies /in Hungarian/, MTA SZTAKI Tanulmányok, 35/1975.

[5] Krámli A., Ruda M. , Hospital morbidity information and query system /in Russian/, Structure and Organization of program packages, Conference, Tbilissi, 1976.

[6] Krámli A., Ratkó I., Ruda M., Soltész J. , Mathematical and computing problems of the statistical data processing /in Hungarian/, MTA SZTAKI Tanulmányok, 7o/1977.

[7] Ratkó I. , Computing and optimizing problems of complicated logikal expressions, /in Hungarian, with English and Russian summary/, MTA SZTAKI Közlemények, 1978.

[8] Ruda M. , A general information system for preparing of the hospital morbidity data system /in Hungarian/, 8. Neumann Conference, Szeged, 1977.

[9] Ruda M. , A generally applicable program optimizing method /in Hungarian, with English and Russian summary/, MTA SZTAKI Közlemények, 1978.

[1o] Ruda M. , A computing method for optimal seting of function tables, with a data processing application /manuscript/.

Ruda Mihály
Computer and Automation Institute Hungarian Academy of Sciences
H 15o2 Budapest, XI. Kende u. 13-17. Hungary

ON THE GENERATION OF BINARY VECTORS BY SOME CLOSED SETS OF BOOLEAN FUNCTIONS (LINEAR FUNCTIONS AND ALTERNATIVES)

Hans-Dietrich Gronau

1. INTRODUCTION AND NOTATION

This is the third paper in a serie of four.

In /2/ the author began studies in the following direction.

Let k be an integer with $k \geq 2$.

Let $V_k = \left\{ \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix} : a_i \in \{0,1\} \quad (i = 1,\ldots,k) \right\}$ and let

$M = \{X_1,\ldots,X_n\} \subseteq V_k$. We define $f(M)$, where f is a Boolean function and $X_i = \begin{pmatrix} a_{i1} \\ \vdots \\ a_{ik} \end{pmatrix}$ for $i = 1,\ldots,n$, by

$f(M) = f(X_1,\ldots,X_n) = \begin{pmatrix} f(a_{11},\ldots,a_{n1}) \\ \vdots \\ f(a_{1k},\ldots,a_{nk}) \end{pmatrix}$. Let K be a set of Boolean functions.

We define the closure $[M]_K$ of M with respect to K.

Definition. Let a sequence $M_K^i \subseteq V_k$ defined by

1^0 $M_K^0 = M$ and

2^0 $M_K^{i+1} = M_K^i \cup \{X: \exists f \in K, \exists X_1,\ldots,X_n \in M_K^i: X = f(X_1,\ldots,X_n)\}$

for $i = 0,1,2,\ldots$

Then let $[M]_K = \lim_{i\to\infty} M_K^i$.

We observe that the successor of M_K^i is a superset of M_K^i and all members of this sequence are subsets of V_k. Hence, this sequence has to be constant starting by some M_K^a. This M_K^a is denoted by $\lim_{i\to\infty} M_K^i$ and by $[M]_K$, accordingly.

We will investigate the following problems.

1. Find K-conditions for M such that M is K-complete, i.e. $[M]_K = V_k$.

2. Find the cardinality of a K-base, i.e. M is K-complete, but any proper subset of M is not K-complete.

In /1/ and /2/ we found sets of functions K for which there are K-bases of different cardinalities. In the cases we will consider here these cardinalities are uniquely.

In /1/ and /2/ we used M_K^1 for the closure of M with respect to K, where K was a closed set of Boolean functions (see /4/). In the first moment the definition given here seems to be quite more general, but in /3/ we will prove $[M]_K = [M]_{[K]}$, where $[K]$ denotes the usual closure of sets of Boolean functions. Moreover, in this paper we will prove $[M]_K = M_K^1$ for closed sets K (Theorem 1). Hence, it is important to study M_K^1 for closed sets K.

In /2/ we chose for K closed sets of selfdual functions, while in /1/ closed sets of nonlinear, nonselfdual and nonmonotonic functions were considered.

In this paper we solve our problems for the following closed sets of Boolean functions. We use the notation by Post (see /4/).

$L_1 = \bigcup_{m \in \mathbb{N}} \langle\!\langle f^m \rangle\!\rangle$ with $f^m(x_1,\ldots,x_m) = c_o + c_1x_1 + \ldots + c_mx_m$, where $c_o,c_1,\ldots,c_m \in \{o,1\}$,

$L_3 = \bigcup_{m \in \mathbb{N}} \langle\!\langle f^m \rangle\!\rangle$ with $f^m(x_1,\ldots,x_m) = c_1x_1 + \ldots + c_mx_m$, where $c_1,\ldots,c_m \in \{o,1\}$ for $m \geq 1$ and $f^o = o$,

$S_1 = \bigcup_{m \in \mathbb{N} \setminus \{o\}} \langle\!\langle f^m \rangle\!\rangle$ with $f^m(x_1,\ldots,x_m) = x_1 \vee x_2 \vee \ldots \vee x_m$,

$S_3 = S_1 \cup \{1\}$, $S_5 = S_1 \cup \{o\}$, $S_6 = S_1 \cup \{o,1\}$,

where $\mathbb{N}$ denotes the set of natural numbers, + denotes the addition modulo 2 and o and 1 denote the constant functions.

These simple functions are perhaps more important for some practical interests.

In /3/ we will finish these studies by solving the problems for all other closed sets of Boolean functions and giving a survey on the results.

2. A THEOREM

Theorem 1. Let $M \subsetneq V_k$ and let K be a closed set of Boolean functions. Then

$$[M]_K = M_K^1 \quad .$$

Proof. $M_K^1 \subseteq [M]_K$ follows by the definition of the closure $[M]_K$. We have to show that equality holds. Assume the contrary. $M_K^1 = M_K^2$ implies $M_K^1 = M_K^i$ for all $i \geq 2$, i.e. $[M]_K = M_K^1$, which is a contradiction. Let $M_K^1 \subset M_K^2$, where the inclusion is used inthe strong sense. Then there is a vector $X \in M_K^2 \setminus M_K^1$.

X was generated from $X_1,\ldots,X_t \in M_K^1$ and a function $f \in K$ by $X = f(X_1,\ldots,X_t)$. By the definition of closed sets K (see /4/) follows, if a function f belongs to K all functions which can be generated from f by addition of fictive variables, belong to K too. Therefore without loss of generality we obtain that the vectors $X_1,\ldots,X_t$ was generated from vectors $X_1',\ldots,X_s' \in M_K^o$ and functions $f_i \in K$ by $X_i = f_i(X_1',\ldots,X_s')$. (If X_i belongs to M, we use the identical function for f_i. Without loss of generality we may assume $id \in K$, because $[M]_K = [M]_K \cup \{id\}$ follows by the definition.)

Hence $\quad X = f(f_1(X_1',\ldots,X_s'),\ldots,f_t(X_1',\ldots,X_s'))$.

Observing that all superpositions of functions of a closed set K belong to K again, there is a function $g \in K$ satisfying $g = f(f_1,\ldots,f_t)$. Thus, $X = g(X_1',\ldots,X_s')$ with $X_1',\ldots,X_s' \in M_K^o$ and $g \in K$, i.e. $X \in M_K^1$, which contradicts our assumption. q.e.d.

3. SOLUTIONS OF THE PROBLEMS

For simplification let $\underline{o},\underline{1},\underline{e}_i$ denote the following vectors of V_k: All components of $\underline{o}$ are o, while all components of $\underline{1}$ are 1. Exactly the i-component of $\underline{e}_i$ is 1, all other are o.

Throughout let $M \subsetneq V_k$.

We will prove the following theorems.

Theorem 2. M is L_1-complete, if and only if M contains at least k-1 vectors $X_1,\ldots,X_{k-1}$ which form jointly with $\underline{1}$ a system of k linearly independent vectors.

Theorem 3. M is L_3-complete, if and only if M contains k linearly independent vectors.

Theorem 4. M is S_1- or S_3-complete, if and only if M contains the vectors $\underline{o},\underline{e}_1,\ldots,\underline{e}_k$.

Theorem 5. M is S_5- or S_6-complete, if and only if M contains the vectors $\underline{e}_1,\ldots,\underline{e}_k$.

By the statements of these theorems we may follow the structure of K-bases, immediately. In particular, we obtain the cardinalities of k-bases.

Corollary 1. Every L_1-base consists of exactly k-1 vectors.

Corollary 2. Every L_3-base consists of exactly k vectors.

Corollary 3. Every S_1- or S_3-base consists of exactly k+1 vectors.

Corollary 4. Every S_5- or S_6-base consists of exactly k vectors.

4. PROOFS

Proof of theorem 3. By the definition of the class L_3 it follows that $[M]_{L_3}$ consists of exactly all vectors which are linear combinations of the vectors of M. Using the wellknown results on vector spaces we obtain the statement of this theorem.

Proof of theorem 2. An immediate consequence of the definitions of the classes L_1 and L_3 is $[M]_{L_1} = [M \cup \{\underline{1}\}]_{L_3}$. Using theorem 3 we get that $M \cup \{\underline{1}\}$ contains k linearly independent vectors $X_1,\ldots,X_k$. If $\underline{1}$ is among them, the theorem follows. If $\underline{1}$ is not among them, there are vectors $X_{i_1},\ldots,X_{i_t}$ $(t \geq 2)$ among them satisfying $\underline{1} = X_{i_1} + \ldots + X_{i_t}$. Then $M \cup \{\underline{1}\}$ contains the system $\{X_1,\ldots,X_k\} \setminus \{X_{i_1}\}$, which has the wished property.

Proof of theorem 4. For every $i \in \{1,\ldots,k\}$ there are vectors $X^i_1,\ldots,X^i_r \in M$ $(r \geq 1)$ with $\underline{e}_i = X^i_1 \vee \ldots \vee X^i_r$. We observe that $X \vee Y$ has exactly in those components a 1, in which at least one of the vectors X and Y has a 1. Hence, $r \leq 2$ and $\{X^i_1, X^i_r\} \leq \{\underline{e}_i, \underline{o}\}$. Of course, $\underline{e}_i \in \{X^i_1, X^i_r\}$. Without loss of generality let $X^i_1 = \underline{e}_i$, i.e. $\underline{e}_i \in M$. In analogy we get $\underline{o} \in M$. In order to show that $\{\underline{o},\underline{e}_1,\ldots,\underline{e}_k\} \leq M$ is sufficient for S_1- or S_3-completeness of M, let us consider an arbitrary vector $\underline{a} \in V_k$, which has 1's exactly in the components $i_1,\ldots,i_s$, where $o \leq s \leq k$, $1 \leq i_1 < \ldots < i_s \leq k$.

Then $\underline{a} = \underline{o}$ for $s = o$ and $\underline{a} = \underline{e}_{i_1} \vee \ldots \vee \underline{e}_{i_s}$ for $s \geq 1$, i.e. $\underline{a} \in [M]_{S_i}$ for $i \in \{1,3\}$.

Proof of theorem 5. By the definition we have
$[M]_{S_5} = [M \cup \{\underline{o}\}]_{S_1}$ and $[M]_{S_6} = [M \cup \{\underline{o}\}]_{S_3}$.
By theorem 4 M is S_5- or S_6-complete, if and only if $\{\underline{o}, \underline{e}_1, \ldots, \underline{e}_k\} \subseteq M \cup \{\underline{o}\}$, from which we obtain the statement of this theorem.

REFERENCES

1. H.-D.Gronau, Erzeugung dualer Vektoren durch gewisse abgeschlossene Mengen Boolescher Funktionen, Rostocker Math. Kolloquium 3 (1977), 45-56.
2. H.-D.Gronau, Erzeugung dualer Vektoren durch selbstduale Funktionen, Wiss. Zeitschrift der Univ. Rostock, Math.-Nat. Reihe 23 (1974), 9, 791-799.
3. H.-D.Gronau, On the generation of binary vectors by Boolean functions, in preparation.
4. S.W.Jablonski, G.P.Gawrilow, W.B.Kudrjawzew, Boolesche Funktionen und Postsche Klassen, Akademie-Verlag Berlin, 197o.

Hans-Dietrich Gronau
Wilhelm-Pieck-Universität Rostock
Sektion Mathematik
DDR-25 Rostock
Universitätsplatz 1

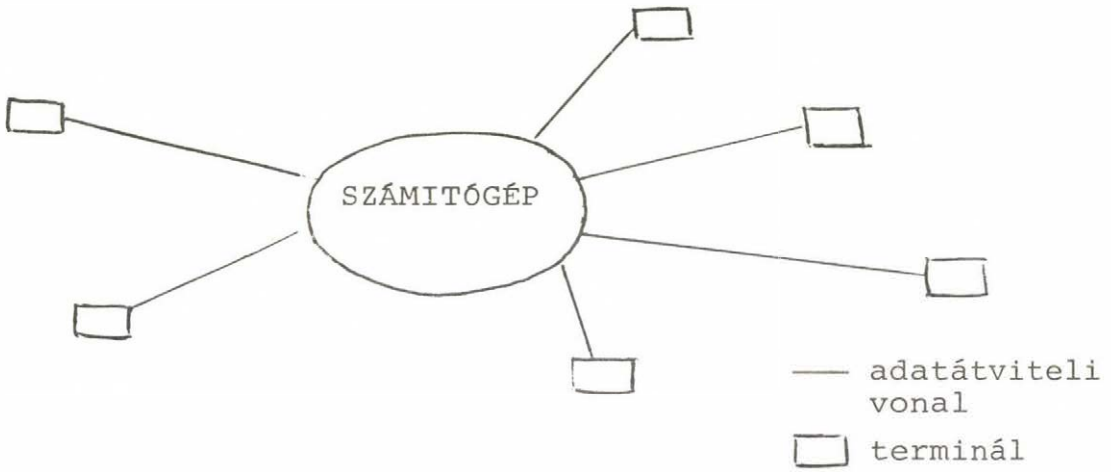
Az MTA tervezett számitógéphálózatával kapcsolatos tervezési szempontokról

Gáspár A.-Kocsis J.-Lamm P.-Visontay Gy.

TARTALOM

1. TERMINÁLHÁLÓZAT, KOMMUNIKÁCIÓS HÁLÓZAT, SZÁMITÓGÉP-HÁLÓZAT

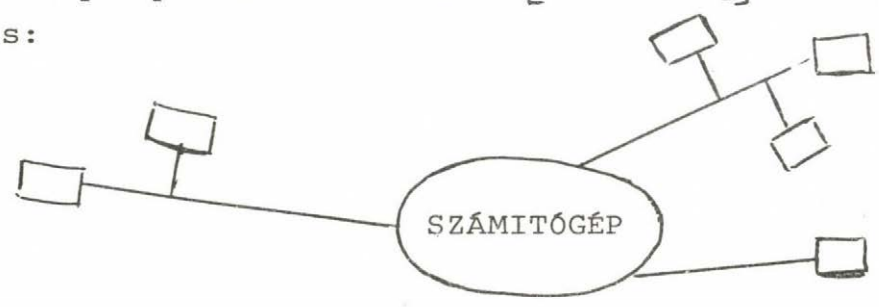
1.1.Körülbelül 15-20 éve rohamos fejlődésnek indultak a terminálhálózatok. Legegyszerübb változatuk a csillaghálózat, mely esetén egy központi számitógéphez néhány távoli terminált kötöttek:



Mivel a távadatfeldolgozás ilyen szervezése nagyszámu, "hosszu" adatátviteli vonalat igényel, elterjedése utjában legalább kétlépcsős "objektiv" akadály áll:

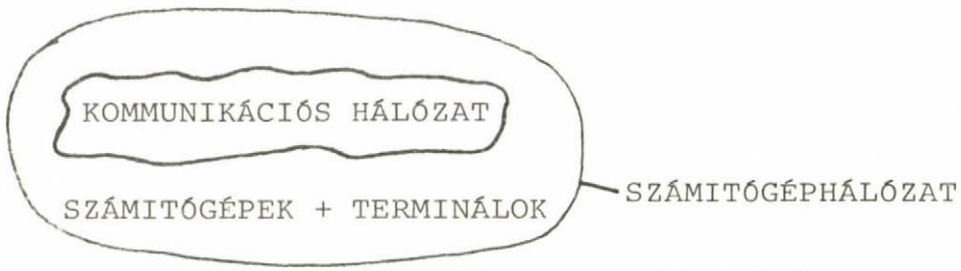
- nincs elegendő adatátviteli vonal,
- már van elegendő adatátviteli vonal, de költsége igen magas.

Lényeges észrevenni azt, hogy a fenti szervezésben az adatátviteli vonalak - éppen a drága erőforrások - általában igen kis mértékben vannak kihasználva. E felismerés egy sereg vonalköltség csökkentő, vonalkihasználtság növelő megoldásra vezetett. Ilyen például a felfüzős [multi-drop] hálózatok szervezése is:



Kezdetben majdnem minden megoldás hardware-megoldás volt, és az a ma már idejétmult elv vezette, mely az adatátviteli vonalak illetve egy-egy adatátviteli hardware-modul kihasználtságának biztositását a központi számitógép feladatának tekintette.

Később a miniszámitógépek árának csökkenésével megjelentek a terminálhálózatok lényegesen rugalmasabb, programozott elemei az adatátviteli funkciók végrehajtására /hibafeltárás, hibajavitás, kódkonverzió, stb./. A távadatfeldolgozás kezdeti szakaszát az igények gyors fejlődése és az adatátvitellel szemben támasztott követelmények kikristályosodása jellemezte. Nyilvánvalóvá vált, hogy valójában két igénycsoport született. Postai--adatátviteli jellegü szolgáltatásra vonatkozik a nyilvános kommunikációs hálózat iránti igény, mig számitástechnikai jellegü a kommunikációs hálózat szolgáltatásait felhasználó számitógéphálózat iránti igény.



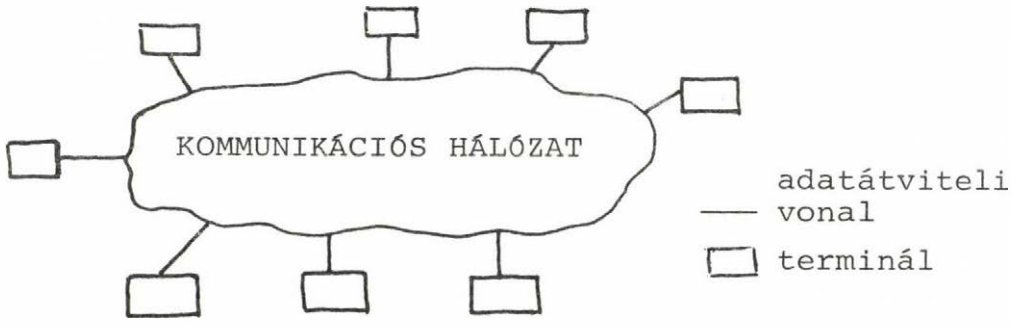
A terminálhálózatok adatátviteli funkcióit megvalósitó kommunikációs hálózatok kifejlesztésének fő céljai a következők:

- csökkenteni az adatátvitel költségeit;
- maximalizálni az adatátviteli vonalak kihasználtságát;
- minimalizálni a felhasznált adatátviteli vonalak számát;
- növelni az adatátvitel biztonságát, sebességét.

Természetesen már a számitástechnikai igények megjelenése előtt is léteztek postai célu kommunikációs hálózatok. Példa rá a telefon-, illetve telexhálózat. Ezzel magyarázható, hogy az ujonnan tervezett hálózatok is megtartották a hagyományos "hivás-kapcsolás" terminológiát. Sajnos az angol terminológia szószerinti magyarra forditása emiatt félrevezető, zavaros

szókincsre vezet. Ezért, egyrészt törekszünk a felhasznált magyar nyelvü fogalmak definiálására, másrészt egy-egy fogalom fontosabb előfordulásai mellett szögletes zárójelben feltüntetjük az angol nyelvü eredetit is.

1.2. Az első igénycsoport tehát egy olyan nyilvános postai szolgáltatásra vonatkozik, mely gyors, nagyteljesitményü, üzembiztos, olcsó elektronikus üzenetváltást biztosit egy hálózat előfizetői között. Ezt a kommunikációs hálózatnak nevezett szolgáltatást a gyorsaság követelménye miatt nagyteljesitményü elemekből szervezik:



A hálózat előfizetői a kommunikációs hálózatot terminálok segitségével használhatják. A terminálok a kommunikációs hálózaton keresztül üzenetnek [message] nevezett hosszu karaktersorozatokat küldenek egymásnak. Egy üzenet tetszőleges számu sorszámmal ellátott levélből [letter] áll.

A kommunikációs hálózatoknak több tipusa alakult ki, melyek főként az üzenettovábbitás szervezésében és teljesitményükben különböznek. Ilyen a vonalkapcsolt [circuit-switched], az üzenetkapcsolt [message-switched], illetve a csomagkapcsolt [packet-switched] kommunikációs hálózat /részletesebben lásd 2./.

A kommunikációs hálózattal szemben támasztott legfontosabb követelmények a következők:

- általános felhasználhatóság /postai, számitástechnikai, stb./;

- gyorsaság /kis átviteli idő/;
- tetszőleges kapcsolat lehetősége /bármely két terminál üzenetet válthasson/;
- megbizhatóság /automatikus hibafeltárás és hibajavitás/;
- nagy kapacitás /nagy adatmennyiség átvitele/;
- magas erőforráskihasználtság;
- alacsony költség.

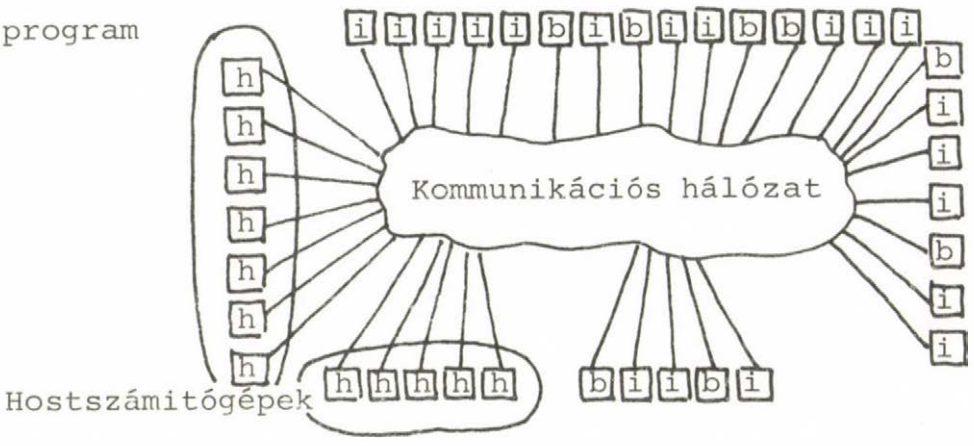
Természetesen a fentiek között ellentmondó követelmények is vannak.

1.3. A második igénycsoport egy olyan különleges kommunikációs hálózatra vonatkozik, mely távadatfeldolgozási célokra számitástechnikai terminálokkal és több - jelentős erőforrásu - központi számitógéppel van összekötve. Az igy felszerelt kommunikációs hálózatot számitógéphálózatnak nevezik.

Számitógéphálózatok termináljainak szerepét általában a következők töltik be:

- interaktiv terminál /például teletype, alfanumerikus display/;
- batch terminál /kártyaolvasó + alfanumerikus display + sornyomtató, batch terminál szimulátor, stb./;
- host program /például az egyik központi erőforrás-számitógép terminál kezelő programja/.

i = interaktiv terminál
b = batch terminál
h = host program



A hagyományos számitógépalkalmazások a bennük domináló számitógépfunkció / futtatás , tárolás/ szerint két nagy csoportot alkotnak. E két alkalmazáscsoport a számitógéphálózatok más-más fejlesztési céljait határozza meg.

1.3.1. A futtatás jellegü alkalmazásokkal kapcsolatos legfontosabb számitógéphálózati célok:

— nagyteljesitményü, gyors, olcsó "távfuttatás" [teleprocessing]
— kivánság szerinti erőforrás - számitógép erőforrásainak elérése

Egy-egy terminálelőfizető /például egy software-ház/ jogos igénye lehet az, hogy ugyanazon terminálról, különféle programjai, különböző számitógépek software, illetve hardware erőforrásait használhassák, /például megfelelő teljesitményü CPU-t, megfelelő operációs rendszert, megfelelő tárkapacitásu mágneslemezt, SIMULA 67 vagy ALGOL 68 compilert, megfelelő adatbáziskezelőt, speciális adatbázisokat, stb./.

- egypéldányu erőforrás fenntartás

  Az erőforrások használata annál olcsóbb, minél több felhasználó között oszlik el a beszerzési és fenntartási költség. A számitógéphálózat költségmegtakaritást jelenthet, mert nagyobb az erőforrás kihasználtság, esetenként elegendő az erőforrást csak egyetlen példányban beszerezni /implementálni/ karbantartani /például SIMULA 67, illetve ALGOL 68 compilerek, speciális adatbázisok/.

- host-számitógépek közötti kommunikáció /főként file átvitel/

  Ha a számitógéphálózat több azonos tipusu host-számitógépet tartalmaz, akkor azok egymás háttérgépei lehetnek és bármelyikük tulterhelése esetén job-okat lehet átirányitani a többihez.

Ha a hálózathoz két különböző, A és B tipusu host-számitógép tartozik, esetleg célszerü egy A gépen müködő, de a B gépre kódot generáló "keresztforditót" implementálni /például egy CDC 3300/R 35 SIMULA forditót/.

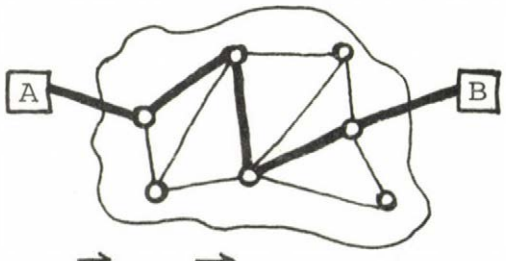
1.3.2. A tárolás jellegü alkalmazásokkal kapcsolatos legfontosabb számitógéphálózat célok:

- gyors, olcsó adatbázislekérdezés és módositás
  Például helyfoglaló rendszerek, megrendeléseket és raktárkészleteket nyilvántartó rendszerek, bibliográfiai adatbankok, egészségügyi adatbankok, stb.
- kivánság szerinti host-számitógép adatbázisainak elérése
  Egy utazási iroda például általában több helyfoglaló rendszert kiván használni.

2. VONALKAPCSOLT, ÜZENETKAPCSOLT ÉS CSOMAGKAPCSOLT KOMMUNIKÁCIÓS HÁLÓZATOK

2.1. A vonalkapcsolt hálózat nagyon hasonló a nyilvános telefonhálózathoz, mely híváskor építi fel a hivó és hivott közötti, az átvitel időtartama alatt rögzitett utat a kapcsológépekben egymáshoz csatlakozó vonalakból. A kapcsolat létrejöttének feltételei a szabad vonalak és terminálok.

□ terminál
○ kapcsológép

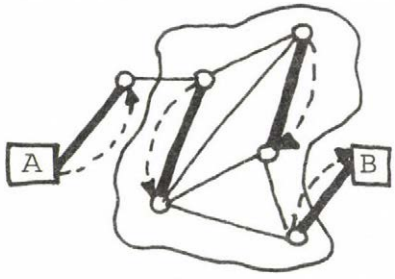


$\overrightarrow{AB}$ és $\overrightarrow{BA}$ forgalom

A kapcsológépekben közbülső adattárolás nincs. Nagymennyiségü adat folyamatos átvitele esetén a vonalkihasználtság jó, egyébként igen alacsony.

2.2. Az üzenetkapcsolt kommunikációs hálózat [message-switched communication system] kapcsológépei ma már általában miniszámitógépek. A kapcsológépeket a hálózat "csucsainak" [node] nevezik. Az üzenetkapcsolt hálózat a forrásterminálról mennyiségi korlátozás nélkül veszi át a cimmel és sorszámmal ellátott, maximált hosszu üzeneteket. Általában max≈4000 karakter. Szükség esetén az átvevő csucs háttértárolóra helyezi őket.

A hálózaton az üzenet csucsról csucsra "ugrik". Mivel az üzenet ugrását egy képzeletbeli "váltóállitás" [switching] előzi meg, ezért a hálózat az üzenetkapcsolt [message-switched] jelzőt kapta. Az üzenet utvonalának közbülső csucsai az üzenetet háttértárolón tartják, mindaddig, amig a következő csucs felé nincs szabad vonalkapacitás, illetve a következő csucsban nincs szabad tárkapacitás a fogadásra/lásd 3. pont/.



$\vec{AB}$ forgalom

□ terminál
○ csucs /miniszámitógép/
--→ üzenet ugrás

Az utazási idő - azaz az üzenet célbaéréséhez szükséges idő - a forgalom függvénye és néhány perctől néhány óráig tarthat.

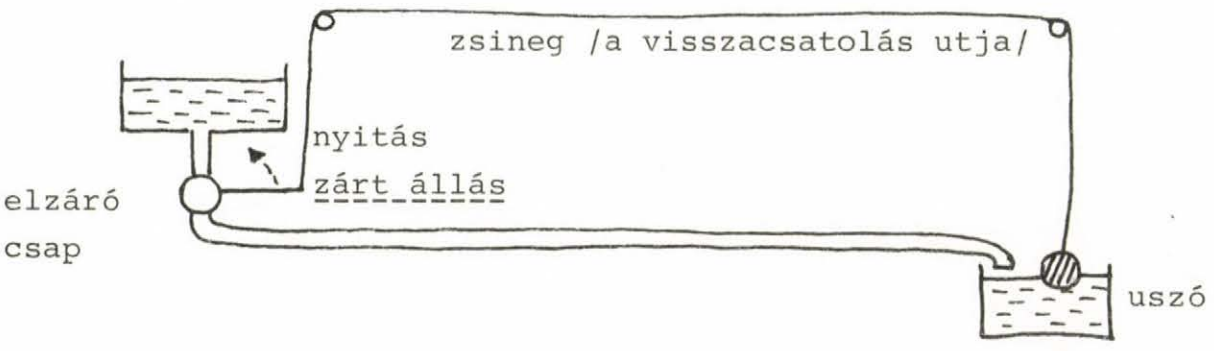
A válaszidő szimmetrikus hálózat, egyenletes forgalom és azonnali válaszadás esetén körülbelül az utazási idő kétszerese.

A hálózat alkalmas táviratok, üzleti levelek továbbitására, hiszen a nappali csucsforgalom idején felvett levelek legfeljebb éjszaka érnek célba. Ezzel szemben, a nagy és bizonytalan válaszidő miatt ez a kommunikációs hálózat interaktiv terminál-számitógép vagy számitógép-számitógép "üzenetváltásra" alkalmatlan.

2.3. A csomagkapcsolt kommunikációs hálózat [packet-switched communication system] hasonló az üzenetkapcsolthoz. Ez is üzeneteket vesz át a forrásterminálttól továbbitásra, azonban a kommunikációs hálózaton mégsem üzenetek "ugrálnak" csucsról csucsra, hanem az üzenet csomagnak [packet] nevezett és sorszámmal ellátott részei /!!/, melyek csak a hálózat határán a célterminál előtti csucsban állnak össze ismét üzenetté. Lényegesebb különbség azonban a következő alapelv: A csomagkapcsolt kommunikációs hálózat a forrásterminálttól a néhány csomagot tartalmazó ugynevezett levelek formájában mindenkor összesen csak annyi csomagot vállal el, amennyit várhatóan igen rövid idő alatt /0.5 - 2.5 sec/ a célterminálba képes juttatni. Igy tehát a leggyakoribb esetben az üzenet [message] eleje már megérkezett a célterminálra, miközben vége még el sem indult a forrásterminálról:

$\vec{AB}$ forgalom

Másfelől viszont, tulterhelés esetén a hálózat a forrásterminálról ritkábban fogad levelet ahelyett, hogy elraktározná a leveleket későbbi szállitásra valamilyen háttértárolón. A csomagkapcsolt hálózat ennek érdekében a célterminál felé kilépő forgalom ismeretében szabályozza a forrásterminálról érkező belépő forgalmat.



A szabályozást a fenti ábra, szükségességét pedig a következő adatok szemléltetik. Egy teletype sebessége 10 karakter/sec, mig egy hostprogram akár 5000 karakter/sec sebességgel is generálhat szállitani valót. A kis terjedési idő követelménye, egy sereg további követelményre vezet. Elsőként emlitendő a névadó csomag kezelés /részletesen lásd 4. pontban/, mellyel elérhető, hogy egy levél csomagjai ne minden csucsban, hanem csakis a célterminál előtti csucsban várják be egymást. A kis terjedési idő követelménye kizárja azt, hogy a közbülső csucsok háttértárolót használhassanak, ezért a továbbitáshoz szükséges erőforrásokra /memória, adatátviteli vonal/ várakozó csomagok a központi memóriában helyezkednek el.

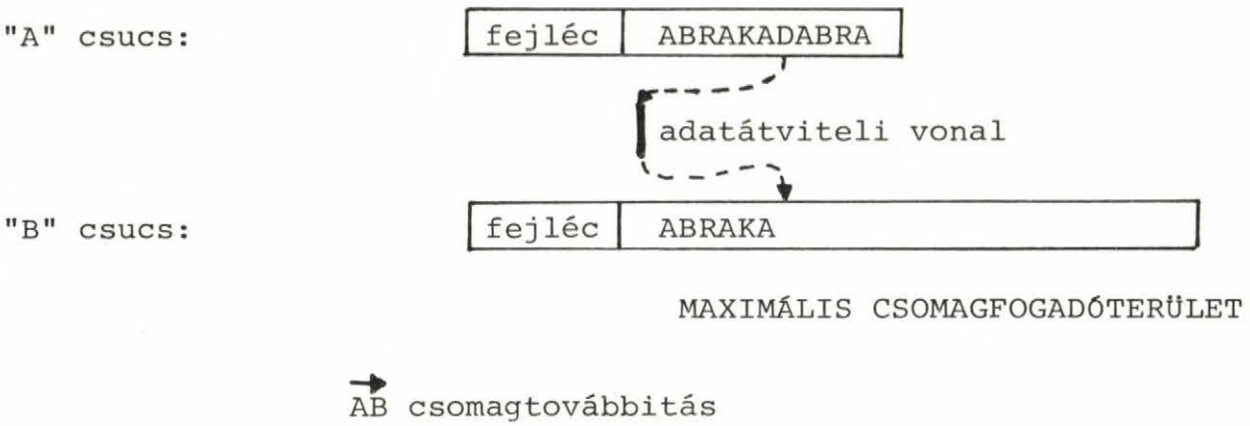
Ugyancsak a kis terjedési idő elérése érdekében a csucsokat nagyteljesitményü vonalakkal kell összekötni, a hálózat rendelkezésére álló kapacitásokat - megfelelő algoritmusok segitségével - "igazságosan" kell elosztani a forrásterminálok között.

3. A CSOMAGKAPCSOLT KOMMUNIKÁCIÓS HÁLÓZAT KÉT SZOMSZÉDOS CSUCSA KÖZÖTTI CSOMAG TOVÁBBITÁS HIBAMENTESSÉGE

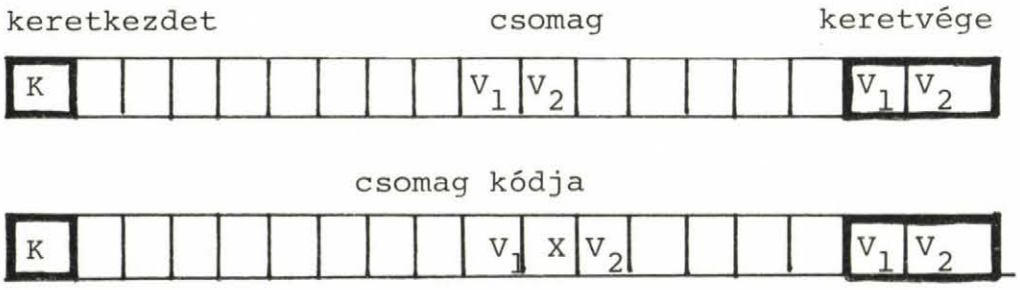
Legyenek A és B szomszédos csucsok és legyen "ABRAKADABRA" az A csucsból a B csucsba továbbitandó csomag tartalma. Ekkor valójában nemcsak az ABRAKADABRA karaktersorozat fog utazni, hanem egy hozzá tartozó fejléc [header] is, mely leirja, hogy a csomag

- melyik forrásterminálról származik;
- melyik célterminál felé tart;
- hányas sorszámu része az üzenetnek;
- hány karakterből áll és mennyi a checksum, stb.

A fejléc hossza általában 96 vagy 192 bit, mert ezek a számok oszthatók a 6, 8, 12, 16, 24, 32 számok mindegyikével. A tartalmat és a fejlécet együtt csomagnak [packet] nevezik.



Az alábbi vonali algoritmus [line protocol] szinkronizálja az A és B csucs müködését. Minden csomag inditását a B csucs kezdeményezi. Lefoglalja a maximális hosszuságu csomag számára szükséges fogadóterületet, majd "fogadásra kész" [receive ready] jelzést küld az A csucsnak. Az A csucsból a csomag egy ugynevezett keretre [frame] helyezve, bitsorozatként utazik a B csucsba. A keretre helyezés azt jelenti, hogy a csomag előtt "keretkezdet" utána pedig "keretvége" jel utazik. Abból a célból, hogy a keret átlátszó legyen, azaz a csomagban előforduló "keretvége" jellel azonos bitsorozat ne keveredjék a valódi "keretvége" jellel, a csomagot a hasonló esetekben szokásos módon kódolják. Például:



A"keretvége" jel megérkezésekor B ellenőrzi, hogy a csomag hibátlan-e, és ha igen, engedélyt ad az A csucsnak az eredeti csomagpéldány eldobására. Ha az átvitel alatt legalább egy bit meghibásodott, akkor a B csucs - igen nagy valószinüséggel - checksum hibát észlel és közli az A csuccsal, hogy a csomagot ujra kell küldeni.

Az adatátviteli vonalak kapacitása nem tetszőleges, hanem olyan diszkrét értékek közül kell választanunk, mint például 2400 bit/sec, 4800 bit/sec, 9600 bit/sec, 48000 bit/sec.

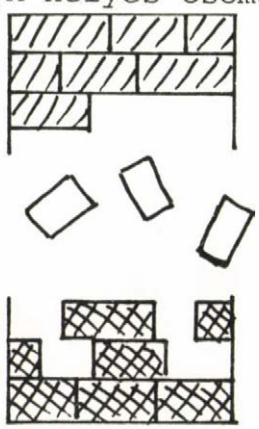
A vonali algoritmus leirásából látszik, hogy a továbbitás alatt álló csomag egyidejüleg két csucsban foglal memóriaterületet. Hosszabb csomag továbbitásakor nyilván hosszabb ideig, és nyilván nagyobb a meghibásodás valószinüsége is. Rövidebb csomagok esetén viszont nagyobbak a fejléc tárolása és szállitása miatti veszteségek. Mekkora tehát végülis a csomagok maximális hosszának optimuma? A különféle csomagkapcsolt hálózatokban a maximális csomagméret általában 128 - 256 byte /1 byte = 8 bit/. A vonali algoritmus feladata mindenfajta vonalhibát lekezelni. A bitmeghibásodás tranziens hiba, de előfordulhatnak permanens hibák is /például vonalszakadás/. Mit tegyen az A csucs, ha a csomag megérkezett a B csucsba, de a B csucsból származó nyugtázás permanens vonalhiba miatt már nem juthatott el az A csucsba? Mindenesetre az A csucs nem tudhatja, hogy mikor következett be a vonalhiba - a csomag, vagy a nyugtázás továbbitása alatt. Az első esetben az A csucsból ujra kellene küldeni a csomagot valamilyen kerülő uton, a második esetben viszont el kellene dobni. Az A csucs rossz döntése esetén csomagdupliká-lódás illetve csomag elveszés történhetne. Mivel a csucsok meghibásodása amugyis csomagelveszésre vezet, célszerü csakis ezt a hiba lehetőséget megengedni.

4. A STARTPONT ÉS CÉLPONT KÖZÖTTI CSOMAG TOVÁBBITÁS HIBAMENTESSÉGE

A csomagkapcsolt kommunikációs hálózatnak a forrásterminállal összekötött csucsát startpontnak [sourcenode], a célterminállal összekötött csucsát pedig célpontnak [destinationnode] nevezik. A startpontba érkező levelet a startpont csomagokra bontja és sorszámuk szerint növekvő sorrendben elinditja őket a célpont felé. A csomagok a hálózat bármely utvonalát hasz-

nálhatják. Igy megfelelő irányitás esetén /lásd 6. pont/ egyrészt elkerülhetik a meghibásodott vonalakat illetve csucsokat és nagyobb a hálózat megbizhatósága, másrészt több egymástól független utvonalat használva ugyanazon levél csomagjai általában hamarabb gyülhetnek össze a célpontban, nagyobb lesz a hálózat teljesitménye.

Mivel a csomagok különböző utvonalat használhatnak, előfordul, hogy a startpontból egymás után induló n, n+1, n+2, n+3, ... sorszámu csomagok egészen más sorrendben érkeznek a célpontba: n+3, n+1, n, n+2,... A helyes csomag sorrend helyreállitását a célpontra bizzák:



Mint azt a 3. pontban láttuk, egy csomag utközben akár el is veszhet. A hiba észleléséről, azonositásáról és kijavitásáról a startpontnak kell gondoskodni, mely minden csomagról egy másolatot tart fenn mindaddig, mig nem értesül annak célbaéréséről a célpontból küldött válasz [response] alapján /96 - 192 bit/. A válasz mindig egy olyan "s" sorszámot hordoz, melyre az s+1 sorszámu csomag még nem ért célba, de minden i≤ s sorszámu már célba ért.

Ha a startpontba már befutott az s sorszámu csomagra küldött válasz, és a startpont már elküldte az s+1 sorszámu csomagot, de még nem kapott rá választ, akkor ez utóbbi csomaggal kapcsolatban 4 eset lehetséges:

- a csomag még uton van;
- a csomag már elveszett;

- a válasz még uton van;
- a válasz már elveszett.

A startpont tehát nem tudhatja, hogy mikor kell a csomagot ujraküldeni, mikor nem fog az ujraküldés csomag duplikálódást okozni. A probléma egyszerüsödik, ha mind a csomagnak, mind a válasznak élettartama van, azaz elküldésük után bizonyos idővel - ha még nem értek célba - akárhol is vannak garantáltan megsemmisülnek. Ha ugyanis élettartamuk rendre T_c illetve T_v, akkor a fenti felételek mellett az s+1 sorszámu csomag elindítása után $T_c + T_v$ idővel már csak 2 eset lehetséges:

- a csomag veszett el;
- a válasz veszett el.

A startpont tehát képes észlelni a hibát, de mint látható, még igy se lehet a célpontbeli duplikálódás veszélye nélkül ujra küldeni a csomagot. A startpontnak a hiba észlelésekor előbb meg kell győződnie arról, hogy valóban a csomag veszett-e el. A hiba azonositása céljából a startpont egy kérdést [enquiry] küld a célpont felé. A célpont a kérdésre megismétli az utoljára elküldött választ. Természetesen a kérdés is elveszhet, és a kérdésnek is élettartama van: T_k. Általában $T_c = T_v = T_k$. Ha a startpontba a kérdés elküldése utáni T_k+T_v idő alatt nem érkezik válasz, akkor 2 eset lehetséges:

- a kérdés veszett el;
- a kérdésre adott válasz veszett el

Ilyen hiba észlelése esetén a startpont még néhányszor megismétli a kérdését, majd ha ezekre sem kapott választ, akkor a kapcsolatot hibásnak nyilvánitja és "elbontja" /lásd 5. pont/.

Ha a startpontba a kérdés elküldése, azaz a t_k időpont után válasz érkezik a t időpontban, mely az s' sorszámot hordozza, akkor:

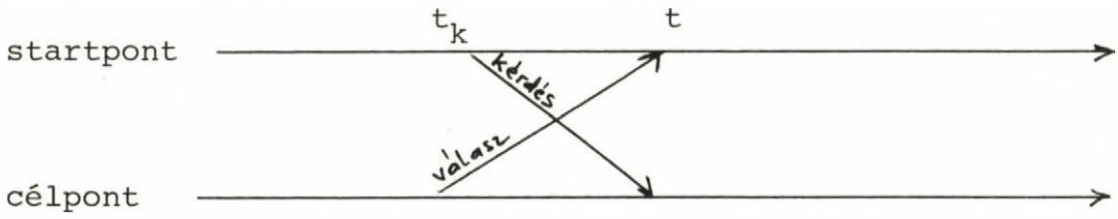
- $s' < s$ esetén csakis "elkésett" válaszról lehet szó, mert a kérdés kibocsájtása után csakis $s' \geq s$ választ

adhat a célpont. Az elkésett választ a startpont eldobja.

- $s' > s$ esetén a startpontnak nem szabad az s+1 sorszámu csomagot ujraküldeni a célpont felé, hiszen az már egyszer célbaért.
- $s' = s$ esetén a befutó válasz a $t-T_v$ időpontnál később indult a célpontból, hiszen különben már megsemmisült volna. Az s+1 sorszámu csomagnak viszont - hacsak nem veszett el - már a t_k-T_v időpontban a célpontban kellett volna lenni. Mivel $t-T_v > t_k-T_v$, az $s'=s$ állapot csakis ugy lehetséges, ha az s+1 sorszámu csomag elveszett. Tehát e csomagot ujra kell küldeni.

A fentiek összefoglalva: A startpont a hiba észlelése és a kérdés kibocsájtása után érkezett válaszok alapján képes a hibát azonositani, egyértelmüen el tudja dönteni, hogy mi a teendő. A startpont az elveszett csomagokat duplikálódás veszélye nélkül pótolni tudja. A startpont tehát képes a hibajavitásra.

A forgalom időviszonyainak szemléltetésére célszerüek a közlekedési menetrendek ábrázolására használt, több időtengelyt tartalmazó ábrák:



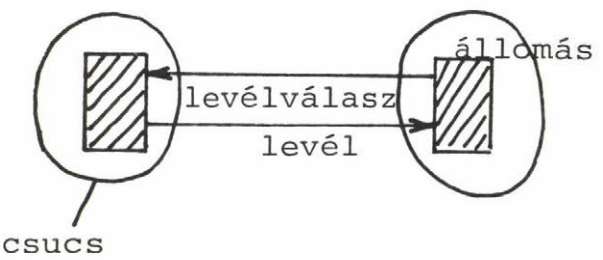
5. A TERMINÁLOK KÖZÖTTI KAPCSOLAT HIBAMENTESSÉGE

Az előfizetői terminálok a csomagkapcsolt kommunikációs hálózatot mint fekete dobozt használják. A hálózat kivülről valóban egy fekete doboz, mely a startpontba érkező leveleket a célpontba és onnan rendre a célterminálra továbbitja. Ideális esetben annyira gyorsan, hogy a terminálok ugy "levelezhetnek"

a hálózaton keresztül, mintha közvetlen /vonalkapcsolt/ kapcsolatuk lenne. Ilyenkor a terminálok észre sem veszik a hálózatot, mert az számukra "átlátszó".

A csomagkapcsolt kommunikációs hálózat a célterminál felé kilépő levélforgalom ismeretében szabályozza a forrásterminálról belépőt: Egyszerre korlátos számu /általában 3-5/ levél lehet uton a forrásterminál és a célterminál között. Amikor a célterminál a célpontnak nyugtázta egy levél megérkezését, akkor a célpont egy levélválaszt küld a startpontba /96-192 bit/. A startpont a levélválaszt feldolgozza. Hatására ujabb levelet vehet át a forrásterminháltól. Természetesen a levélválasz is elveszhet, ezért róla másodpéldányt kell fenntartani. Mivel elveszését figyelemmel kell kisérni, és elveszése esetén ujra kell küldeni, a levélválasz pontosan olyan kezelést kiván, mint az ellentétes irányu kapcsolat /célterminál forrásterminál/ csomagjai. A legcélszerübb tehát, ha az ellentétes irányu kapcsolat egy speciális csomagjaként kezelik.

Egy kapcsolat startponti funkcióit és az ellentétes irányu kapcsolat célponti funkcióit megvalósitó programokat és adatokat együtt állomásnak [transport station] nevezik. Az állomáson csomagok, illetve levelek "tartózkodnak".

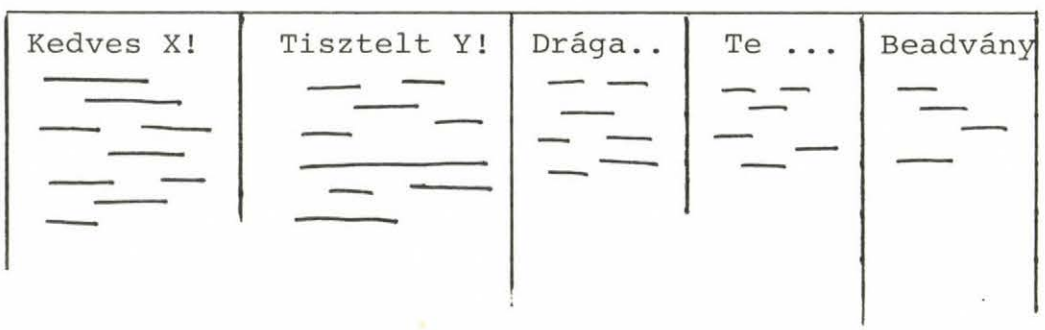


Egy-egy kapcsolat fenntartása nemcsak az adatátviteli vonalakat terheli, hanem jelentős memóriát is igényel a következő célokra:

- a levelek fogadására /a startpontban/, sorbaállitására és továbbitására /a célpontban/;

- a csomag másolatok megőrzésére /a startpontban/, csomagok tárolására és továbbitására /a csucsokban/, illetve összegyüjtésére /a célpontban/.

Vagyis az állomásokon tartózkodó küldeményeknek jelentős memóriaigénye van. Igy tehát, ha egy uj kapcsolat a szükséges memória előzetes biztositása nélkül léphetne üzembe, akkor az erőforrásokat elvehetné a már üzemelő kapcsolatoktól, tulterhelést és üzemzavart okozva /részletesebben lásd 6. pont/.



Egy célpontban a teljes memóriát levéltöredékek foglalhatnák le, vagyis patthelyzet alakulhatna ki

Minden kapcsolat használata előtt tehát egy erőforrásokat lefoglaló-ellenőrző folyamat szükséges. E folyamat neve hivás [virtual call]. Sikeres híváskor a két terminál között egy kapcsolatpár jön létre és továbbitásra kész állapotba kerülnek az állomások.

Ellentétes folyamat a bontás. Ekkor megszünik a kapcsolatpárt kezelő állomások továbbitóképessége és elvesznek a megfelelő állomásokon tartózkodó vagy oda később befutó levelek, illetve csomagok. Ez a folyamat egy erőforrásokat felszabaditó folyamat.

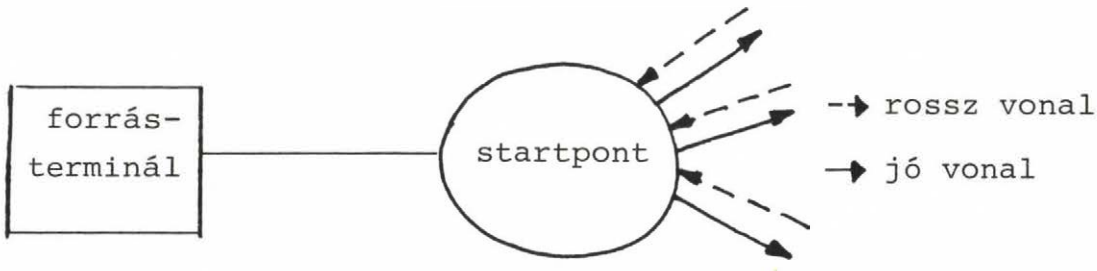
A hivási folyamatot az egyik terminál, mint forrásterminál kezdeményezi. Ha startpontjában nem áll rendelkezésre a kapcsolat fenntartásához szükséges memória, a hivási folyamat sikertele-

nül befejeződik. Ha a memória rendelkezésre áll, a startpont hivást bocsájt ki a célpont felé. A hivásnak is élettartama van, mert a hivás is elveszhet. Ha hosszu ideig nem érkezik hivásválasz a startpontba, akkor a hivási folyamatot a startpont sikertelenül befejezettnek tekinti.Természetesen a hivásválasznak is van élettartama és az is elveszhet.

Ha a hivás eljutott a célpontba, akkor a célpont megkisérli lefoglalni a kapcsolatpár fenntartásához szükséges erőforrásokat /terminál, memória, állomás, stb./.

A kisérlet eredményét a hivásválaszban közli a startponttal és sikeres erőforráslefoglalás esetén a kapcsolatpárt felépitettnek tekinti. Ha a startpontba befutott hivásválasz szerint a kapcsolatpár felépitése sikertelen volt, a hivási folyamat befejeződik. A hivási folyamat mindenfajta sikertelen befejezése esetén a startpont megszünteti az állomás továbbitóképességét, felszabaditja a lefoglalt memóriát és meghatározott formátumu levélben "közli" a forrásterminállal a sikertelenség okát.

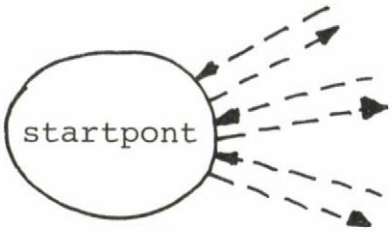
Tételezzük fel a következő helyzetet:



A fenti helyzetben valamennyi hivás sikertelen lesz, mert a startpontba nem juthat el semmilyen hivásválasz. Ha a kiszemelt terminál rendre felhivná a hálózat összes többi terminálját /az ideges felhasználó esete/ és azok kedvező hivásválaszt adnának, akkor a fenti terminál időtlen időkre lefoglalná /magához rendelné/ a hálózat valamennyi terminálját. A hálózat megbénulna.

Az időtlen időkre szóló hozzárendelés veszélye máskor is fennáll.

Tegyük fel, hogy egy már müködő kapcsolatban a startpont vonalhibák miatt "időtlen időkre" elszigetelődik a hálózat többi részétől:



Bizonyos idő elteltével a startpont megkezdi a kérdések kibocsájtását és mivel a kérdésekre "a célpont nem válaszol" megkezdi a kapcsolat elbontását /lásd 4. pont/; megszünteti a startpontbeli állomás továbbitóképességét; eldobja az ott tartózkodó leveleket, csomagokat; felszabaditja az általuk lefoglalt memóriát. Ha ilyenkor a startpont értesitést küldene az elbontásról, az éppen ugy elveszne, mint a kérdések.

Tehát időtlen időkre müködésben maradna az elbontandó kapcsolat célpontjabeli állomás és lefoglalva maradnának az ott lekötött erőforrások.

A vázolt problémák megoldására szolgálnak a tétlen kapcsolatfenntartás és az automatikus kapcsolatbontás szabályai.

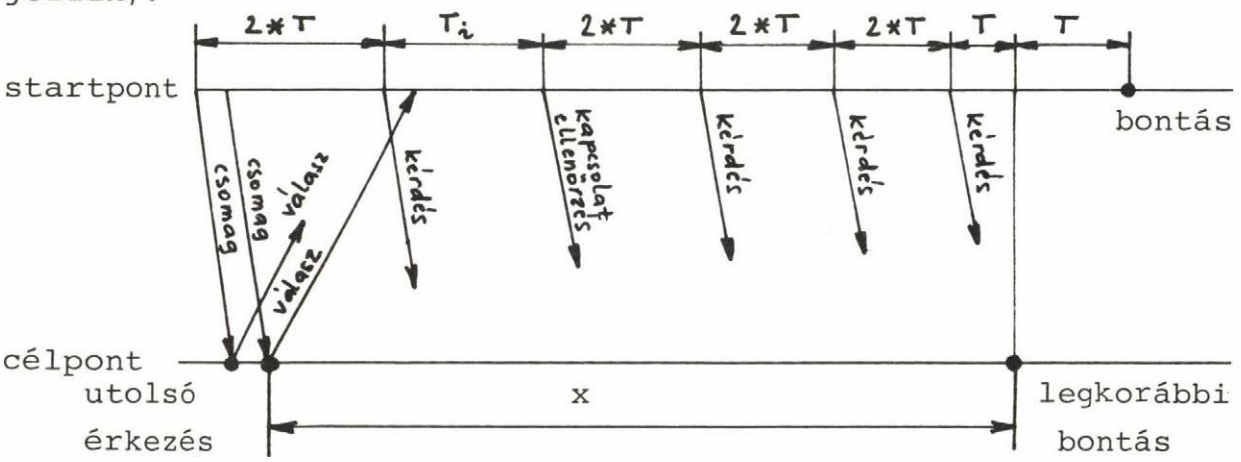
Az állomások startpont funkciót ellátó része kapcsolatellenőrzést [idle] küld a célpont felé, ha az állomást a célpont felé már "régóta" /T_i/ nem hagyta el semmi /hivás, hivásválasz, csomag, válasz, kérdés, levélválasz, kapcsolatellenőrzés/. Természetesen a kapcsolatellenőrzésnek is élettartama van /T_e/ és elveszhet. Mind a startpont, mind a célpont a kapcsolatellenőrzést egy speciális kérdésként kezeli /elveszés esetén "ujabb" kérdés megy, célbajutás esetén megismétli az utolsó választ, stb./

E feltételek mellett, ha egy állomás célpont funkciót ellátó részébe elegendően hosszu ideig nem érkezett csomag, kérdés, illetve kapcsolatellenőrzés, akkor már nem is érkezhet, mert mindannyian elvesztek utközben és emiatt a startpont már bontotta is a

kapcsolatpárt.

Ekkor tehát a célpont is bonthat.

Legyen $T_k = T_v = T_e = T$. Ha a startpont n számu sikertelen kérdés után kezdi meg a bontást, akkor mennyi idő mulva bonthat legkorábban automatikusan a célpont? Tegyük fel, hogy ez az érték x. A legrosszabb eset a következő /az ábrán a startpontot vagy célpontot el nem érő nyilak az elveszett küldeményeket jelzik/:



$$x = T + T_i + 2*T + (n-1)*2*T + T = T_i + 2*(n+1) * T$$

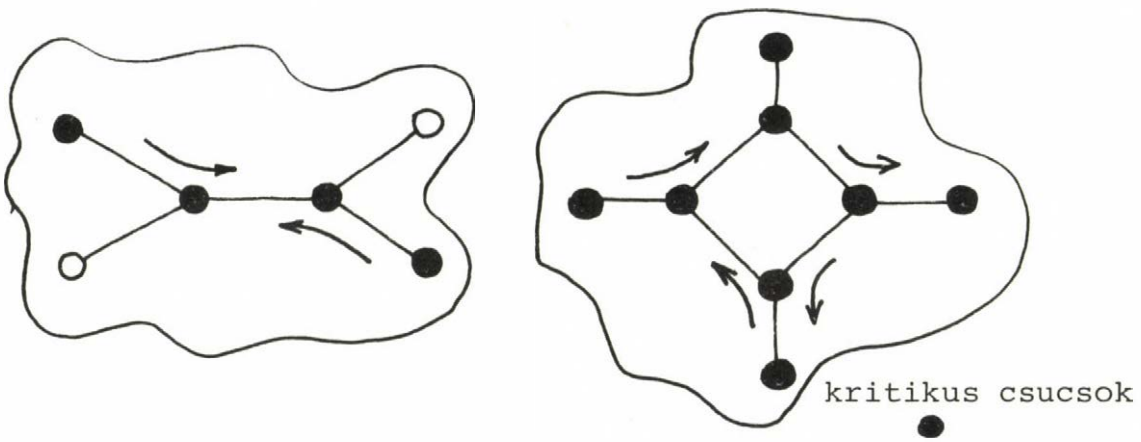
A fenti érték nemcsak az automatikus bontás időpontját határozza meg, hanem a bontást követő hivás illetve a sikertelen hivást követő hivás legkorábbi időpontját is.

6. A KOMMUNIKÁCIÓS HÁLÓZAT MÜKÖDÉSÉNEK PATTHELYZET-MENTESSÉGE

A csomagkapcsolt kommunikációs hálózat, mint láttuk, nemcsak csomagokat, hanem hivást, hivásválaszt, választ, kérdést, kapcsolatellenőrzést és levélválaszt is továbbit. Mindezek - akárcsak a közönséges csomagok - fejléccel ellátva - speciális csomagként közlekednek. Utközben teljesen lényegtelen e csomagok tartalma, érdekessé csakis utazásuk célpontjában válik. Fontos analógiára jutunk, ha a hálózatot városi uthálózatként,

a csomagokat pedig rajta közlekedő autókként képzeljük el. A városi közlekedés jellegzetes hibái a patthelyzetek, a torlódások, illetve a közlekedési dugók.

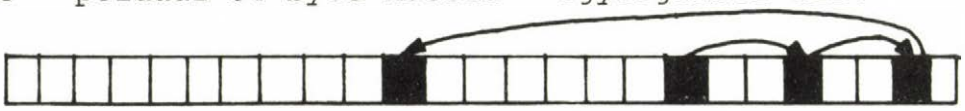
A kommunikációs hálózaton is felléphetnek hasonló patthelyzetek:



A fent ábrázolt hálózatokon a nyillal jelölt csomagforgalmak egymást akadályozzák, mert a kritikus csucsokban nincs fogadóterület /memória/ a forgalom fenntartására.

A patthelyzetek vizsgálata céljából memóriaanalizisre lesz szükség. Tekintsük át először a csucsbeli memóriakezelés rendjét.

A csucsokban a vonalak fogadóterületei, a továbbításra képes állomások és a tranzitcsomagok kötnek le memóriát tárolás céljára. A tárolóterületként használt memória tetszőlegesen láncra füzhető - például 64 byte hosszu - egységekből áll:

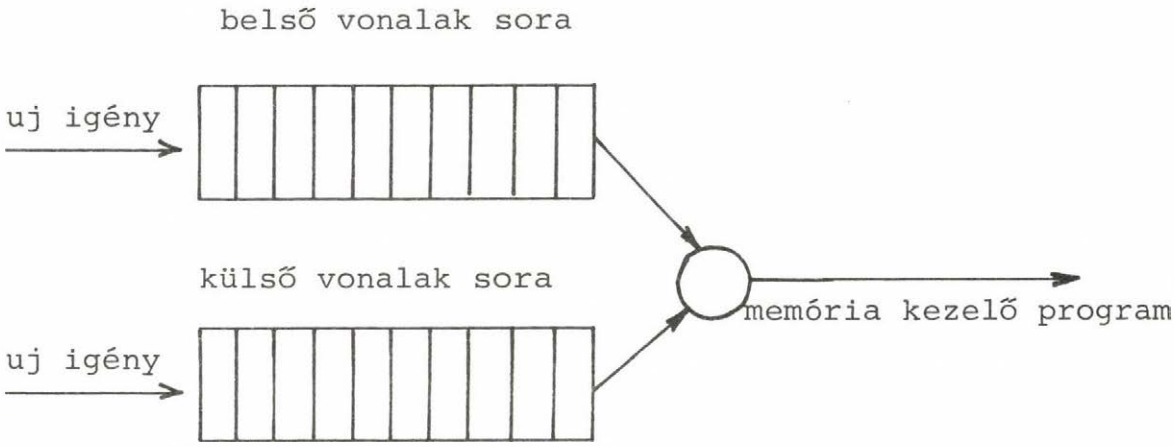


egy fogadóterület

Egy csomag lefoglalhat például 1, 2, 3, 4 vagy 5 ilyen memóriaegységből álló láncot. Mivel az adatátviteli berendezések képesek a láncolt átvitelre, ezért semmi szükség sincs az átvitel előtt egy olyan másolásra, mely a láncot folytonossá tenné. Ugyancsak a láncolt memóriaszervezés és adatátvitel lehetősége miatt a beérkező levelek csomagra bontása, illetve a célpontban

a csomagok levéllé összeállitása nem igényel külön memóriát. A hálózat belső vonalai a csucsok közötti vonalak, külső vonalai pedig a terminálok és csucsok közötti vonalak. A külső vonalakon levélforgalom, a belső vonalakon pedig csomagforgalom zajlik. A külső vonalak használatát az állomások terminálkezelő programjai [terminal process], a belső vonalak használatát pedig a csucsok vonal kezelő programjai [link process] szervezik /fogadó terület foglalás, forrás terminálok periódikus poll-ozása, "fogadásra kész" jelzés, stb./

Az adatátviteli vonalak sorbanállnak a fogadóterületért /memóriáért/:



A hálózat belső vonalainak prioritása nagyobb, mint a külső vonalaké. Egy-egy soron belül a kiszolgálás az igények érkezési sorrendjében történik /FIFO/.

A belső vonalak számára a maximális csomagméret, a külső vonalak számára a maximális levélméret határozza meg a fogadóterület nagyságát. Ha a beérkezett csomag, illetve levél kisebb méretü, mint a számára előzőleg lefoglalt fogadóterület, akkor a fogadóterületből a memória bizonyos számu egysége felszabadul.

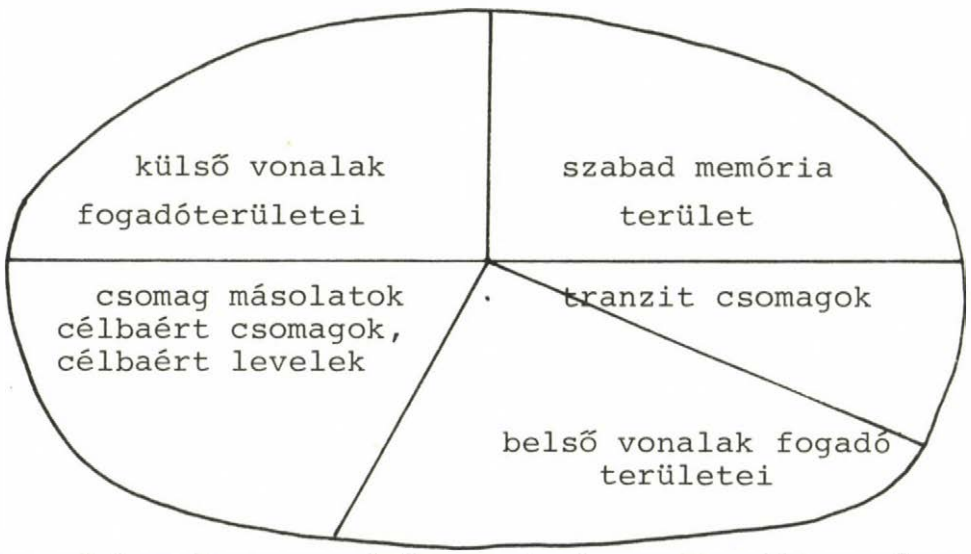
Memória szabadul fel akkor is, ha a csucsból tranzitcsomag, vagy levél távozott.

Az állomásokon memóriát foglalnak az elküldött csomagok másolatai, az összeállitásra várakozó csomagok, a célpontból a célterminálra továbbitásra várakozó levelek. Befutó válaszok hatására az állomás esetleg csomag másolatokat enged el.

Az állomások memóriaigénnyel lépnek fel a következő csomagok elküldésekor: hivás, hivásválasz, válasz, kérdés, kapcsolatellenőrzés. Ugyanilyen tipusu csomagok célbaérésekor a fogadó állomás felszabaditja az általuk lefoglalt memóriát.

Ha elküldésükkor nem állna rendelkezésre a küldő állomás részére a kivánt nagyságu memória, akkor fenti csomagokat az állomás nem tudja elküldeni, de mégis elküldöttnek, később pedig elveszettnek tekinti őket. A fent emlitett tipusu csomagok memóriakezelés szempontjából elküldésük után a küldőcsucsban pontosan ugy kezelhetők, mintha tranzit csomagok lennének.

A fentiek szerint minden csucsban az ott lefoglalt memóriát egy adott pillanatban az alábbi ábrával szemléltethetjük:



A hálózat lehetséges patthelyzeteinek egyik változatát az előző pontban mutattuk be. Abban az esetben csupa félig kész levél foglalta le valamely csucsban a teljes memóriát [reassembly lock-up]. E jelenség kizárására a csucs korlátos memóriaterületet tart fenn az állomások és a külső vonalak fogadóterületei számára, továbbá, e megengedett terület tullépése elleni véde-

kezésül minden kapcsolatpárban legfeljebb annyi levél lehet uton a terminálok között, mint amennyiben a híváskor a startpont és a célpont megállapodik /hivás, hivásválasz/.

Tegyük fel a következőket:

- a csucsban üzemkezdetkor a szétoszthtató memóriaterület mérete M memóriaegység;
- a csucstól egyidejüleg legfeljebb k darab fogadóterületet igényelhetnek a csucs belső vonalai;
- a csucs legfeljebb m egységnyi memória területet tart fenn az állomások és a külső vonalak fogadóterülete számára.

Előfordulhat-e ekkor patthelyzet, ha m+k * maximális csomagméret << M? A fenti feltételek teljesülése mellett patthelyzet esetén csakis a tranzitcsomagok akadályozhatnák a csucsba irányuló forgalom fenntartását. Csakhogy a tranzitcsomagok által lekötött memória rövid idő alatt felszabadul, mert a tranzitcsomag vagy

- távozik a csucsból, vagy
- megsemmisül, mert élettartama lejár.

A felszabaduló memóriaterületből a memóriakezelés előbb vázolt szabályai szerint előbb-utóbb bármelyik, fogadóterületre várakozó belső vonal részesül. Vagyis egyetlen irányban se bénul meg véglegesen a forgalom. Patthelyzetről tehát nem beszélhetünk, legfeljebb csak lényeges teljesitményromlást okozó torlódásról.

7. A CSOMAGKAPCSOLT KOMMUNIKÁCIÓS HÁLÓZAT HIBATŰRŐKÉPESSÉGE

Az előző pontokban vázoltuk a csomagkapcsolt kommunikációs hálózat hibamentes müködéséhez szükséges küldemények tipusait és a kommunikációs hálózat rétegeit. Szemléltetésükre a követ-

kező oldalon látható ábrát javasoljuk.

Egy számitástechnikai rendszer hibatürőképessége azt jelenti, hogy a rendszer inputjában, komponenseiben, illetve komponensei kapcsolatában fellépő hibák ellenére is képes eredeti funkcióját ellátni, a hibák ellenére is megbizhatóan müködik. Egy hierarchikus rendszer esetében ehhez a rendszer minden rétegében megfelelő hibaészlelés, hibaazonositás, illetve hibajavitás szükséges. Az előző pontok szerint a csomagkapcsolt kommunikációs hálózat tökéletesen hibatürő, ha egy rövid ideig tartó hiba bekövetkezése után elegendően hosszu ideig nincsen ujabb hiba.

Permanens csucshiba esetén a csucs elszigetelődik szomszédaitól: nem fogad és nem küld ujabb csomagot. Ezért előbb a szomszédos csucsokban hosszu várakozási sorok keletkeznek, majd később az elszigetelődött csucson lévő állomások párjai kapcsolatot bontanak.

Egy adatátviteli vonal permanens hibája esetén - ha az ujraküldött csomag beállna a permanensen hibás vonal előtti várakozási sorba - az elveszett csomagok pótlása is kudarcba fulladna.

Ezért tehát a permanensen hibás komponensek /csucsok, vonalak/ előtti várakozási sort meg kell szüntetni, a csomagokat kerülő utvonalra kell irányitani. Kerülő utvonal persze nem minden topológia és nem minden kapcsolat esetén létezik.

Az adaptiv utvonalkijelölés igénye azonban nemcsak a hibás komponensek elkerülése, a hibatürőképesség növelése miatt lép fel. Ugyanerre a következtetésre jutunk, ha a patthelyzetek, illetve torlódások elkerülése, vagy a teljesitménynövelés a cél.

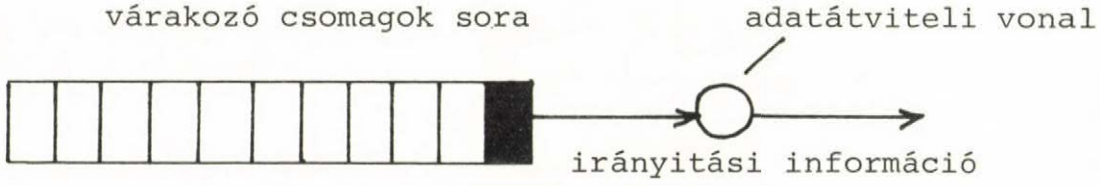
Mielőtt vázolnánk az adaptiv utvonalkijelölést, tekintsük át az utvonalkijelölést általában. Az utvonalkijelölés lépésről-lépésre történik, a csomagot csucsról csucsra irányitják a célja fe-

KOMMUNIKÁCIÓS HÁLÓZAT
levél
üzenet
/magasszintü/ hivásválasz
/magasszintü/ hivás
Host/
user
Host/
user
forrásterminál
célterminál
levélkezelés
levélválasz
kapcsolatellenőrzés
állomás
/startpont/
kérdés
válasz
csomag
hivásválasz
hivás
állomás
/célpont/
csomag-
kezelés
küldő csucs
hibás
keret [frame]
fogad.kész
fogadó csucs
ÜZENETTOVÁBBITÁSI PROTOKOLL
(filetransfer)
LEVÉLTOVÁBB. PROTOKOLL
CSOMAGTOVÁBBIT. PROTOKOLL
VONALI PROT.

lé. A kommunikációs hálózat minden csucsának van egy-egy utvonal táblázata, /irányitási táblázat/ mely az irányitást vezérli.

Az irányitás lehet a hálózat topológiájával lerögzitett, vagy adaptiv, azaz a csucson kivülről érkező információhoz alkalmazkodó. Joggal felmerülő kérdés: Lehetséges-e központi matematikai irányitás? A kérdést felvető bizonyára a közuti közlekedés számitógépes irányitására gondol példaként. Csakhogy a közlekedésben a forgalmi helyzet változásának sebessége az utvonalkijelöléshez szükséges irányitási információ terjedési sebességéhez /rádió/ viszonyitva kicsi, elhanyagolható.

Nem ez a helyzet a kommunikációs hálózatban. Az adaptiv irányitáshoz szükséges információ is a hálózaton terjed csucsról csucsra. Az irányitási információ [update-packet] két szomszédos csucs között utazó csomag, mely a továbbitásra várakozó csomagok sorának elejére áll, megsemmisitve az esetleg már ott várakozó elődjét:



Az irányitási információ tehát pontosan olyan gyorsan terjed, mint a többi csomag. Központi matematikai irányitás tehát azért lehetetlen, mert müveletei végrehajtásának időtartama alatt /információgyüjtés, elemzés, parancstovábbitás/ a hálózat állapota lényegesen változhat.

Az irányitási algoritmus tehát csak olyan információt használhat fel, amely az irányitás pillanatában már helyileg rendelkezésére áll. Vagyis az adaptiv irányitáshoz a csucsok irányitási táblázatát kell rendszeresen módositani.

A szomszéd csucsoknak küldött irányitási információ egy távolságvektor, mely leirja a küldő csucsba érkező csomag várható

továbbutazási idejét az összes lehetséges célpont felé. Az irányitási információ küldhető periódikusan, illetve csakis lényeges állapotváltozás hatására. E távolságvektor a csucsbeli irányitási táblázat pillanatnyi állapotából és a várakozási sorok hosszából becsülhető.

Adaptiv irányitás esetén egy-egy csucs irányitási táblázata egy olyan távolságmátrix, mely minden célponthoz és a csucs minden "kijáratához" hozzárendeli a rákövetkező csucstól várható továbbutazási időt.

Ilymódon az adaptiv irányitás a távolságmátrix és a "kijáratok" előtt álló sorok ismeretében a csomagot a legrövidebb utvonalra irányitja. A csomagok - a hálózat megfelelő topológiája esetén - képesek a meghibásodott komponenseket kikerülni.

8. TOVÁBBI ALAPPROBLÉMÁK

A csomagkapcsolt kommunikációs hálózatok, illetve az ezekre épülő számitógéphálózatok esetében egy sereg további olyan problémával kerülünk szembe, melyek részletes vizsgálata meghaladja jelen irásunk kereteit. Ezért itt csupán cimszavakban történő felsorolásukra és vázolásukra vállalkozhatunk.

a/ Csomagok ciklikus sorszámozásának problémája
A küldött és fogadott csomagok sorszámozását mod N végezzük, ami azt jelenti, hogy a csomagsorszámok N elérése után ismétlődnek. Erre azért van szükség, mert a csomag fejlécében korlátos méretü helyen tárolható a csomagsorszám. E döntés egyik következménye, hogy tilos a startpontból egy az N-től és a minimális utazási időtől függő korlátnál több, még nyugtázatlan csomagot utjára inditani. Ugyanis ellenkező esetben a ciklikus sorszámozás csomagkeveredéshez vezethetne.

b/ A szabványositás problémája

E téren jelentős nemzetközi erőfeszitések vannak, azonban nehéz megegyezéseket elérni, mivel nagyértékü meglévő nemzeti/ /magán beruházások kerülnének ezzel "szabványon kivülre". A szabványositás igénye nemcsak a hardware-re és az illesztésre terjed ki, hanem a müködést szabályozó algoritmusokra, protokollokra is.

A csomagkapcsolt hálózatokra vonatkozó legjelentősebb szabványositási eredménynek a CCITT; 25-ös szabványa tekinthető. Ha a magánhálózatok fejlődést akadályozó sokfélesége, inkompatibilitása helyett megvalósulhatna a nemzetközi szabvány szerinti nyilvános hálózatok rendszere, akkor a hálózatok az uj hardware és software erőforrások /pl. ILLIAC IV, ALGOL XX/ gyors elterjesztésében kiemelkedő szerepet játszhatnának.

c/ A magasszintü protokollok problémája

A magasszintü protokollok erősen függenek az erőforrásszámitógépek hardware-jétől, operációs rendszerétől. Gondot okoz szabványositásuk.

d/ A hálózatok minőségvizsgálatának problémája

A valós hálózat mérési költségei rendkivül magasak, a nagyszámu szétszórt mérési pont, a mérőeszközök szükséges szinkronizációja, illetve a mérés beavatkozásmentességének biztositása miatt. Ráadásul a valós hálózaton zajló folyamatok gyakorlatilag megismételhetetlenek, ezért az alternativ algoritmusok, paraméterek vizsgálata is nehézkes.

A hálózatok matematikai vizsgálatára rendkivül nagy igény lenne, azonban a bonyolultság miatt nem sok valóban használható eredményre számithatunk. Adekvát vizsgálati módszernek tehát a szimuláció látszik.

IRODALOM

1 Kleinrock, L.: Queuing Systems. Vol 2, Computer Applications Wiley and Sons, 1967

2 Davies, D.W.: Flow Control and Congestion Control, COMNET 77, Budapest, Working Papers, Vol 1, 1977

3 Price, W.L.: Data Network Simulation, Experiments at the National Physical Laboratory, 1968-76 Computer Networks I, North Holland Publ., 1977

4 Davies-Barber: Számitógéphálózatok. Müszaki Könyvkiadó, 1978

5 Gáspár A.-Kocsis J.-Lamm P.-Visontay Gy.: Az MTA tervezett számitógép-hálózatának minőségvizsgálata Információ Elektronika, 1978/4

6 CCITT x 25 Recommendation

7 Szentiványi T.-Tallóczy I.: Computer Networks, SZÁMKI Közlemények 17, Budapest, 1978

Gáspár András
Kocsis József
Lamm Péter
Visontay György

MTA Számitástechnikai és Automatizálási Kutató Intézet
1111 Budapest, Kende utca 13-17.

S Z Á M O K REPRO

367/1978